

Tight Bounds for the Distribution-Free Testing of Monotone Conjunctions

Xi Chen

Columbia University

xichen@cs.columbia.edu

Jinyu Xie

Columbia University

jinyu@cs.columbia.edu

Abstract

We improve both upper and lower bounds for the distribution-free testing of monotone conjunctions. Given oracle access to an unknown Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and sampling oracle access to an unknown distribution \mathcal{D} over $\{0, 1\}^n$, we present an $\tilde{O}(n^{1/3}/\epsilon^5)$ -query algorithm that tests whether f is a monotone conjunction versus ϵ -far from any monotone conjunction with respect to \mathcal{D} . This improves the previous best upper bound of $\tilde{O}(n^{1/2}/\epsilon)$ by Dolev and Ron [DR11] when $1/\epsilon$ is small compared to n . For some constant $\epsilon_0 > 0$, we also prove a lower bound of $\tilde{\Omega}(n^{1/3})$ for the query complexity, improving the previous best lower bound of $\tilde{\Omega}(n^{1/5})$ by Glasner and Servedio [GS09]. Our upper and lower bounds are tight, up to a poly-logarithmic factor, when the distance parameter ϵ is a constant. Furthermore, the same upper and lower bounds can be extended to the distribution-free testing of general conjunctions, and the lower bound can be extended to that of decision lists and linear threshold functions.

1 Introduction

The field of property testing analyzes the resources an algorithm requires to determine whether an unknown object satisfies a certain property versus *far* from satisfying the property. It was introduced in [RS96], after prior work in [BFL91, BLR93], and has been studied extensively during the past two decades (see surveys in [Gol98, Fis01, Ron01, AS05, Rub06]).

For our purpose, consider a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and a class \mathcal{C} of Boolean functions, viewed as a property. The distance between f and \mathcal{C} in the standard testing model is measured with respect to the *uniform distribution*. Equivalently, it is the smallest fraction of entries of f one needs to flip to make it a member of \mathcal{C} . A natural generalization of the standard model, called *distribution-free property testing*, was first introduced by Goldreich, Goldwasser and Ron [GGR98] and has been studied in [AC06, HK07, GS09, HK08a, HK08b, DR11]. In the distribution-free model, there is an unknown distribution \mathcal{D} over $\{0, 1\}^n$ in addition to the unknown f . The goal of an algorithm is to determine whether f is in \mathcal{C} versus far from \mathcal{C} *with respect to \mathcal{D}* , given black-box access to f and sampling access to \mathcal{D} . The model of distribution-free property testing is well motivated by scenarios where the distance being of interest is indeed measured with respect to an unknown distribution \mathcal{D} . It is also inspired by similar models in computational learning theory (e.g., the distribution-free PAC learning model [Val84] with membership queries). It was observed [GGR98] that any proper distribution-free PAC learning algorithm can be used for distribution-free property testing.

In this paper we study the distribution-free testing of *monotone conjunctions* (or *monotone monomials*): f is a monotone conjunction if $f(z) = \bigwedge_{i \in S} z_i$, for some $S \subseteq [n]$. We first obtain an efficient algorithm that is one-sided and makes $\tilde{O}((n^{1/3}/\epsilon^5))$ queries. When $1/\epsilon$ is small compared to n , it improves the previous best $\tilde{O}(n^{1/2}/\epsilon)$ -query algorithm of Dolev and Ron [DR11].

Theorem 1.1. *There is a $O((n^{1/3}/\epsilon^5) \cdot \log^7(n/\epsilon))$ -query one-sided algorithm for the distribution-free testing of monotone conjunctions.*

For some constant distance parameter $\epsilon_0 > 0$, we also present a $\tilde{\Omega}(n^{1/3})$ lower bound on the number of queries required by any distribution-free testing algorithm. This improves the previous best lower bound of $\tilde{\Omega}(n^{1/5})$ by Glasner and Servedio [GS09].

Theorem 1.2. *There exists a universal constant $\epsilon_0 > 0$ such that any two-sided distribution-free algorithm for testing whether an unknown Boolean function is a monotone conjunction versus ϵ_0 -far from monotone conjunctions with respect to an unknown distribution must make $\Omega(n^{1/3}/\log^3 n)$ queries.*

Notably when the distance parameter ϵ is a constant, our new upper and lower bounds given in Theorems 1.1 and 1.2 are tight for the distribution-free testing of monotone conjunctions up to a poly-logarithmic factor of n . Furthermore, these bounds can also be extended to several other basic Boolean function classes.

First, our upper bound can be extended to general conjunctions (i.e. f is the conjunction of a subset of literals in $\{z_1, \dots, z_n, \bar{z}_1, \dots, \bar{z}_n\}$) via a reduction to the distribution-free testing of monotone conjunctions, improving the previous best $\tilde{O}(n^{1/2}/\epsilon)$ -query algorithm of Dolev and Ron [DR11] when $1/\epsilon$ is small.

Theorem 1.3. *There is a $O((n^{1/3}/\epsilon^5) \cdot \log^7(n/\epsilon))$ -query one-sided algorithm for the distribution-free testing of general conjunctions.*

Second, our lower bound can be extended to the distribution-free testing of general conjunctions, decision lists, as well as linear threshold functions (see their definitions in Section 2), improving the previous best lower bound of $\tilde{\Omega}(n^{1/5})$ by Glasner and Servedio [GS09] for these classes. For general conjunctions, our bounds are also tight up to a poly-logarithmic factor of n when ϵ is a constant.

Theorem 1.4. *There exists a universal constant $\epsilon_0 > 0$ such that any two-sided distribution-free algorithm for testing whether an unknown Boolean function is a general conjunction versus ϵ_0 -far from general conjunctions with respect to an unknown distribution must make $\Omega(n^{1/3}/\log^3 n)$ queries. The same lower bound holds for testing decision lists and testing linear threshold functions.*

In most part of the paper we focus on the distribution-free testing of monotone conjunctions (except for Sections 5, 6 and 7). We start with some intuition behind our new algorithm and lower bound construction for monotone conjunctions, and compare our approaches and techniques with those of [GS09] and [DR11].

1.1 The Lower Bound Approach

We start with the lower bound because our algorithm was indeed inspired by obstacles we encountered when attempting to push it further to match the upper bound of Dolev and Ron [DR11].

We follow the same high-level approach of Glasner and Servedio [GS09]. They define two distributions \mathcal{YES} and \mathcal{NO} : in each pair (f, \mathcal{D}_f) drawn from \mathcal{YES} , f is a monotone conjunction, whereas in each (g, \mathcal{D}_g) drawn from \mathcal{NO} , g is constant-far from monotone conjunctions with respect to \mathcal{D}_g . Then they show that no algorithm with $\tilde{O}(n^{1/5})$ queries can distinguish them. We briefly review their construction and arguments.

Both distributions start by sampling $m = n^{2/5}$ pairwise disjoint sets C_i of size $n^{2/5}$ each. Each C_i is then randomly partitioned into two disjoint sets A_i, B_i of the same size, with a special index α_i randomly sampled from A_i . Let a^i, b^i, c^i denote the strings with $A_i = \text{ZERO}(a^i)$, $B_i = \text{ZERO}(b^i)$, and $C_i = \text{ZERO}(c^i)$, where we write $\text{ZERO}(x) = \{i : x_i = 0\}$. For \mathcal{YES} , f is the conjunction of x_{α_i} 's, $i \in [m]$, and x_j 's, $j \notin \cup_i C_i$. So $f(a^i) = f(c^i) = 0$ and $f(b^i) = 1$. \mathcal{D}_f puts weight $2/(3m)$ on b^i and $1/(3m)$ on c^i . The definition of \mathcal{NO} is much more involved. g sets $g(a^i) = g(b^i) = 1$ and $g(c^i) = 0$; \mathcal{D}_g is uniform over all $3m$ strings $\{a^i, b^i, c^i\}$. On the one hand, g is clearly far from monotone conjunctions with respect to \mathcal{D}_g . On the other hand, by the birthday paradox, any algorithm that draws $n^{1/5}$ samples with high probability gets at most one sample from each triple (a^i, b^i, c^i) , and information theoretically cannot distinguish \mathcal{YES} and \mathcal{NO} : What the algorithm sees is just a bunch of pairwise disjoint sets of two sizes, as $\text{ZERO}(x)$ of samples x received. In discussion below we refer to them as the sets the algorithm receives in the sampling phase¹.

The real challenge for Glasner and Servedio is to define g in \mathcal{NO} carefully on strings of $\{0, 1\}^n$ outside of $\{a^i, b^i, c^i\}$ such that even an algorithm with access to a black-box oracle cannot distinguish them. For this purpose, g follows f by setting $g(x) = 0$ whenever $x_j = 0$ for some $j \notin \cup_i C_i$. This essentially discourages a reasonable algorithm from querying z with $z_j = 0$ for some j outside of the sets it received in the sampling phase: for any such z , both f and g return 0 with probability $1 - n^{1/5}$ so with only $n^{1/5}$ queries the risk is too high to take. Knowing that an algorithm only queries such strings, [GS09] sets up g so that an algorithm can distinguish g from f only when it incurs an event that is unlikely to happen (e.g., hitting $z_{\alpha_i} = 0$ in some A_i with a query z that has a small $\text{ZERO}(z) \cap A_i$). When events like this do not happen, the algorithm can be successfully simulated with no access to the black-box oracle. This finishes the proof.

Our lower bound proof follows similar steps as those of Glasner and Servedio [GS09]. *The improvement mainly comes from a more delicate construction of the two distributions \mathcal{YES} and \mathcal{NO} , as well as a tighter analysis on a no-black-box-query simulation of any testing algorithm with access to both oracles.* The first difficulty we encountered is a dilemma in the construction: There are only n indices in total but we want the following three things to happen at the same time: We need $n^{2/3}$ sets C_i 's so that the birthday paradox still applies for $n^{1/3}$ queries; We would like each C_i to have size $n^{2/3}$ to survive black-box queries; Also $\cup_i C_i$ is better small compared to n so one can still argue that no reasonable algorithm makes any crazy black-box

¹Without loss of generality, we may always assume that an algorithm starts by a sampling phase when it receives all the samples drawn from \mathcal{D} . After that it only queries the black-box oracle.

query z with zero entries outside of the sets it receives. There is simply no way to satisfy all these conditions; Glasner and Servedio had the best parameters in place and they are tight in more than one places.

It seems that the only possible solution is to allow C_i 's to have significant overlap with each other. This, however, makes the analysis more challenging, since an algorithm may potentially gain crucial information from intersections of sets it receives in the sampling phase. Informally we first randomly pick a set R of size $n/2$ and randomly partition it into $n^{1/3}$ disjoint blocks of size $n^{2/3}$ each. Each of the $n^{2/3}$ sets C_i 's consists of $2 \log^2 n$ random blocks and two special indices α_i and β_i that are unique to C_i . Each C_i is then partitioned into A_i, B_i with $\log^2 n$ blocks each, which also receive α_i and β_i , respectively. An important property from our setup (and simple calculation) that is crucial to our analysis later on is that even with $\tilde{O}(n^{1/3})$ sets drawn uniformly, most likely only a $o(1)$ -fraction of each set is covered by other sets sampled. The rest of our \mathcal{YES} and \mathcal{NO} is similar to [GS09], but with a more intricate \mathcal{NO} function g outside of the support of \mathcal{D}_g .

Our distributions \mathcal{YES} and \mathcal{NO} work well against any algorithm with no access to the black-box oracle. The technically most challenging part is to show that any given algorithm can be simulated closely without the black-box oracle. Note that $\cup_i C_i$ above is about $n/2$. An algorithm with $n^{1/3}$ many queries has a much stronger incentive to take the risk and query z with $z_j = 0$, for some j outside of the sets sampled. This then demands a more sophisticated analysis to characterize every possible loophole an algorithm may explore, in distinguishing the two distributions \mathcal{YES} and \mathcal{NO} . At the end, we need to fine-tune the construction of \mathcal{NO} to really fit the analysis perfectly (not surprising given the upper bound) so that we can manage to bound the probability of each loophole, and show that the no-black-box-query simulation succeeds most of the time.

1.2 The Approach of Our Algorithm

We now describe the high-level approach of our algorithm. For clarity, we assume here that ϵ is a constant. We first review the $\tilde{O}(n^{1/2})$ -query algorithm of Dolev and Ron [DR11]. An ingredient from [DR11], which we also use heavily as a subroutine, is a deterministic binary search procedure: upon $x \in f^{-1}(0)$, it attempts to find an index $i \in \text{ZERO}(x)$ such that $f(\{i\}) = 0$.² If it fails on x , then f is not a monotone conjunction; otherwise, let $h(x)$ denote the index found, called the representative index of x [DR11]. Roughly speaking, the algorithm of Dolev and Ron draws $n^{1/2}$ samples from \mathcal{D} and uses the binary search procedure to compute the representative index $h(x)$ of each sample x from $f^{-1}(0)$. Then the algorithm rejects if $y_\alpha = 0$ for some sample $y \in f^{-1}(1)$ and some representative index α found. The algorithm is one-sided. But to reject with high probability when f is far from monotone conjunctions with respect to \mathcal{D} , $n^{1/2}$ samples seem necessary.

Our algorithm was inspired by obstacles encountered when trying to improve the $\tilde{\Omega}(n^{1/3})$ lower bound. To give some intuition, consider the same distribution of triples of sets, (A_i, B_i, C_i) , drawn as in the lower bound proof sketch, with $m \approx n^{2/3}$ many C_i 's each of size $\approx n^{2/3}$. Let \mathcal{D} be the uniform distribution over $\{a^i, b^i, c^i\}$, with g satisfying $g(a^i) = g(b^i) = 1$ and $g(c^i) = 0$. Now consider the following scenario where an adversary tries to fill in entries of g outside of $\{a^i, b^i, c^i\}$, aiming to fool algorithms with a small number of queries as a monotone conjunction. An obstacle for the adversary is the following testers: Let $t \approx n^{1/3}$.

Tester 1. Draw t samples y^1, \dots, y^t from $g^{-1}(1)$ with respect to \mathcal{D} . Let $E_i = \text{ZERO}(y^i)$, $E = \cup_i E_i$. Given the definition of \mathcal{D} and that $g(a^i) = g(b^i) = 1$ and $g(c^i) = 0$, each E_i is either A_k or B_k . Repeat t times: pick a subset Z of E of size t uniformly at random and query z with $\text{ZERO}(z) = Z$. (Note that if g is a monotone conjunction, then E cannot contain any index of a variable that belongs to the conjunction and hence for every $Z \subseteq E$ and z with $\text{ZERO}(z) = Z$, g must return 1 to query z .)

Tester 2. Draw $t - 1$ samples y^1, \dots, y^{t-1} from $g^{-1}(1)$ with respect to \mathcal{D} , and one sample x from $g^{-1}(0)$ (so $\text{ZERO}(x) = C_k$ for some k). Define E_i and E similarly. Use the binary search procedure

²For convenience we extend f to subsets of $[n]$, with $f(A)$ defined as $f(z)$ with $A = \text{ZERO}(z)$.

to find the representative index $h(x)$ of x ; for the sake of discussion here assume that it finds the special index α_k in C_k if $\text{ZERO}(x) = C_k$ (reject if $\alpha_k \in E$). Pick a subset Z of E of size $t - 1$ uniformly at random, and query z with $\text{ZERO}(z) = Z \cup \{\alpha_k\}$. (Note that if g is a monotone conjunction, then $h(x)$ must be the index of a variable in the conjunction and hence, we have $h(x) \notin E$ and for every $Z \subseteq E$ and z with $\text{ZERO}(z) = Z \cup \{h(x)\}$, g must return 0 to query z .)

Consider an algorithm that runs both testers with independent samples. Clearly g fails and gets rejected if it returns 0 to a query z from Tester 1 or it returns 1 to a query z from Tester 2. It turns out that there is no way to design a g that returns the correct bit most of the time for both testers. To see this is the case, assume for now that about half of the E_i 's in Tester 1 are indeed A_i 's so each of them contains a special and unique index α_i ; in total there are $\Omega(t)$ many of them in E . Given that $|E| \leq n$, and we repeat t times in picking z , most likely one of the strings z queried has an $\alpha_i \in \text{ZERO}(z)$ and it is also the only index in $\text{ZERO}(z) \cap E_i^*$, where we let E_i^* denote the indices that are unique to E_i among all E_j 's. (For the latter, the intuition is that there simply cannot be too many large E_i^* because they are disjoint and their union is E .)

For such a string z drawn and queried in Tester 1, g has to return 1. However, the distribution of such z is very similar to the distribution of z queried in Tester 2, where an α_k is first picked randomly (by drawing a C_k and running the binary search procedure on it to reveal α_k) and then unioned with a set of $t - 1$ indices drawn uniformly from E obtained from $t - 1$ samples from $g^{-1}(1)$.

This is essentially how our algorithm works. It consists of two stages, each of which implements one of the two testers. The main challenge for us is the analysis to show that it works for any input pair (f, \mathcal{D}) that not necessarily looks like those constructed in \mathcal{NO} . At a high level, we show that if f is far from monotone conjunctions with respect to \mathcal{D} and passes stage 1 with high probability, then it fails stage 2 and gets rejected with high probability since the two distributions of z queried in the two stages are very close to each other.

An important ingredient of our analysis is the notion of a *violation bipartite graph* G_f of a pair (f, \mathcal{D}) . Compared to the *violation hypergraph* H_f introduced by Dolev and Ron, our bipartite graph G_f is easier to work with and its vertex covers also characterize the distance between f and monotone conjunctions (similar to the violation hypergraph of [DR11]). In particular, our analysis of correctness heavily relies on a *highly regular* bipartite subgraph G_f^* of G_f , of which every vertex cover still has total weight $\Omega(\epsilon)$. The regularity of G_f^* plays a critical role in our comparison of the two stages. More specifically, it helps bound the double counting when we lower bound the probability of (f, \mathcal{D}) failing stage 2, assuming that it passes stage 1 with high probability.

Organization. We define the model of distribution-free testing, and introduce some useful notation in Section 2. We present the new algorithm for monotone conjunctions and its analysis in Section 3, followed by the lower bound proof in Section 4. We then extend the upper bound to general conjunctions in Section 5, and extend the lower bound to general conjunctions and decision lists in Section 6, and to linear threshold functions in Section 7.

2 Preliminaries

We review the model of distribution-free property testing and then introduce some useful notation.

Let $f, g : \{0, 1\}^n \rightarrow \{0, 1\}$ denote two Boolean functions over n variables, and \mathcal{D} denote a probability distribution over $\{0, 1\}^n$. We define the distance between f and g with respect to \mathcal{D} as

$$\text{dist}_{\mathcal{D}}(f, g) = \Pr_{z \in \mathcal{D}} [f(z) \neq g(z)].$$

Given a class \mathfrak{C} of Boolean functions over $\{0, 1\}^n$, we define

$$\text{dist}_{\mathcal{D}}(f, \mathfrak{C}) = \min_{g \in \mathfrak{C}} \left(\text{dist}_{\mathcal{D}}(f, g) \right)$$

as the distance between f and \mathfrak{C} with respect to \mathcal{D} . We also say f is ϵ -far from \mathfrak{C} with respect to \mathcal{D} for some $\epsilon \geq 0$ if $\text{dist}_{\mathcal{D}}(f, \mathfrak{C}) \geq \epsilon$. Now we define distribution-free testing algorithms.

Definition 2.1. Let \mathfrak{C} be a class of Boolean functions over $\{0, 1\}^n$. A distribution-free testing algorithm T for \mathfrak{C} is a probabilistic oracle machine with access to a pair (f, \mathcal{D}) , where f is an unknown Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and \mathcal{D} is an unknown probability distribution over $\{0, 1\}^n$, via

1. a black-box oracle that returns the value $f(z)$ when $z \in \{0, 1\}^n$ is queried; and
2. a sampling oracle that returns a pair $(z, f(z))$ with z drawn independently from \mathcal{D} each time.

The algorithm T takes as input a distance parameter $\epsilon > 0$ and satisfies for any (f, \mathcal{D}) :

1. If $f \in \mathfrak{C}$, then T accepts with probability at least $2/3$; and
2. If f is ϵ -far from \mathfrak{C} with respect to \mathcal{D} , then T rejects with probability at least $2/3$.

We say an algorithm is one-sided if it always accepts a function f in \mathfrak{C} .

In this paper we focus on the distribution-free testing of MCONJ , the class of all monotone conjunctions (or monotone monomials as in [DR11]): $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is in MCONJ if there exists an $S \subseteq [n]$ with

$$f(z_1, \dots, z_n) = \bigwedge_{i \in S} z_i.$$

Note that f is the all-1 function when S is empty. In addition to monotone conjunctions we are interested in the distribution-free testing of general conjunctions, decision lists, and linear threshold functions:

- We say $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is a general conjunction if there exist two sets $S, S' \subseteq [n]$ with

$$f(z_1, \dots, z_n) = \left(\bigwedge_{i \in S} z_i \right) \wedge \left(\bigwedge_{i \in S'} \bar{z}_i \right).$$

- A decision list $f : \{0, 1\}^n \rightarrow \{0, 1\}$ of length k over Boolean variables z_1, \dots, z_n is defined by a sequence of k pairs $(\ell_1, \beta_1), \dots, (\ell_k, \beta_k)$ and a bit β_{k+1} , where $\beta_i \in \{0, 1\}$ for all $i \in [k+1]$ and each ℓ_i is a literal in $\{z_1, \dots, z_n, \bar{z}_1, \dots, \bar{z}_n\}$. Given any $z \in \{0, 1\}^n$, $f(z)$ is determined in the following way: $f(z) = \beta_i$ if $i \in [k]$ is the smallest index such that ℓ_i is made true by z ; if no ℓ_i is true then $f(z) = \beta_{k+1}$.
- We say $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is a linear threshold function if there exist $w_1, w_2, \dots, w_n, \theta \in \mathbb{R}$ such that $f(z) = 1$ if $w_1 z_1 + \dots + w_n z_n \geq \theta$ and $f(z) = 0$ if $w_1 z_1 + \dots + w_n z_n < \theta$.

Next we introduce some notation used in the proofs. Given a positive integer n we let $[n] = \{1, \dots, n\}$. Given a distribution \mathcal{D} over $\{0, 1\}^n$ we use $\mathcal{D}(z)$ to denote the probability of a string z in $\{0, 1\}^n$ and $\mathcal{D}(C)$ to denote the total probability of strings in $C \subseteq \{0, 1\}^n$.

We call x a 0-string (with respect to f) if $f(x) = 0$, and write $f^{-1}(0)$ to denote the set of 0-strings. We call y a 1-string (with respect to f) if $f(y) = 1$, and write $f^{-1}(1)$ to denote the set of 1-strings.

For both our lower and upper bound proofs, it is easier to use the language of sets. Given $z \in \{0, 1\}^n$:

Algorithm 1: Binary Search. Input: $x \in f^{-1}(0)$.

1. Let $Z = \text{ZERO}(x)$. If $Z = \emptyset$, return nil; if $|Z| = 1$, output the only index in Z .
2. While $|Z| \geq 2$ do
 - Let Z_0 be the subset of Z that contains the smallest $\lceil |Z|/2 \rceil$ indices in Z , and $Z_1 = Z \setminus Z_0$.
 - Query both $f(Z_0)$ and $f(Z_1)$.
 - If $f(Z_0) = 0$, set $Z = Z_0$; if $f(Z_0) = 1$ but $f(Z_1) = 0$, set $Z = Z_1$; otherwise, return nil.
3. Return the only element that remains in Z .

Figure 1: The binary search procedure from Dolev and Ron [DR11].

$$\text{ZERO}(z) = \{i \in [n] : z_i = 0\}.$$

For convenience we abuse the notation and allow f to take as input a subset of $[n]$: $f(E)$ is defined as $f(z)$ with $z \in \{0, 1\}^n$ and $E = \text{ZERO}(z)$. This should be clear from the context, since we use lowercase letters for strings and uppercase letters for sets. We call A a 0-set if $f(A) = 0$, and B a 1-set if $f(B) = 1$.

We use $\mathbf{1}^n$ to denote the all-1 string of length n and drop the n when it is clear from the context.

3 Upper Bound: Proof of Theorem 1.1

In this section, we present our one-sided distribution-free tester for MCONJ. Throughout the section we use $f : \{0, 1\}^n \rightarrow \{0, 1\}$ to denote the unknown Boolean function, and \mathcal{D} to denote the unknown distribution.

For clarity of the analysis in this section, we always write x to denote a string from $f^{-1}(0)$, y to denote a string from $f^{-1}(1)$, and z to denote a string with $f(z)$ unknown (or we do not care about $f(z)$).

3.1 Binary Search, Empty Strings, and Representative Indices

The algorithm of Dolev and Ron [DR11] uses a deterministic binary search procedure which, given a string $x \in f^{-1}(0)$, tries to find an index $i \in \text{ZERO}(x)$ such that $f(\{i\}) = 0$. (Note that such an i always exists if f is in MCONJ.) Our algorithm also uses it as a subroutine so we include it in Figure 1 for completeness.

We record the following property of the binary search procedure:

Property 3.1. *The binary search procedure uses $O(\log n)$ many queries. Given as an input $x \in f^{-1}(0)$, it returns either nil or an index $i \in \text{ZERO}(x)$ such that $f(\{i\}) = 0$. The former never happens if $f \in \text{MCONJ}$.*

Given $x \in f^{-1}(0)$, we write $h(x) \in [n] \cup \{\text{nil}\}$ to denote the output of the binary search procedure on x ($h(\cdot)$ is well-defined since the procedure is deterministic). We follow [DR11] and call $x \in f^{-1}(0)$ an *empty* string (with respect to f) if $h(x) = \text{nil}$, and call $h(x) \in [n]$ the *representative index* of x (with respect to f) when $h(x) \neq \text{nil}$.

3.2 A One-sided Algorithm for Testing Monotone Conjunctions

We use the following parameters in the algorithm and its analysis:

$$d = \frac{\log^2(n/\epsilon)}{\epsilon}, \quad d^* = d^2/\epsilon, \quad r = n^{1/3}, \quad t = d \cdot r \quad \text{and} \quad s = t \log n. \quad (1)$$

Our algorithm is presented in Figure 2, which consists of three stages. We refer to it as *Algorithm 2* and start its analysis with the following simple observations.

Algorithm 2: Monotone Conjunctions.

Stage 0. Query $f(1^n)$ and **Reject** if $f(1^n) = 0$. Make $3t(d^* + 1)/\epsilon$ many queries to the sampling oracle. Let $(z^{i,j}, f(z^{i,j}))$ denote the pairs received, for $i \in [d^* + 1]$ and $j \in [3t/\epsilon]$. Run the binary search procedure to compute the representative index $h(x)$ for each $x \in f^{-1}(0)$ sampled. **Reject** if one of them has $h(x) = \text{nil}$.

Stage 1. **Accept** if the number of $j \in [3t/\epsilon]$ with $z^{1,j} \in f^{-1}(1)$ is less than t ; otherwise, we let y^1, \dots, y^t be the first t (not necessarily distinct) 1-strings in $(z^{1,j})$. Let $B_i = \text{ZERO}(y^i)$, $B = \cup_i B_i$.

1.1. Repeat s times: Draw an index i from B uniformly at random. **Reject** if $f(\{i\}) = 0$.

1.2. Repeat s times: Draw a subset $Z \subseteq B$ of size r uniformly at random. **Reject** if $f(Z) = 0$.

Stage 2. Repeat the following steps for d^* iterations. For the i th iteration, $i \in [d^*]$:

Accept if the number of $j \in [3t/\epsilon]$ with $z^{i+1,j} \in f^{-1}(1)$ is less than $t - 1$ or no string in $(z^{i+1,j})$ is from $f^{-1}(0)$; otherwise, let y^1, \dots, y^{t-1} be the first $t - 1$ (not necessarily distinct) 1-strings from $(z^{i+1,j})$, and x be the first 0-string from $(z^{i+1,j})$. Let $B_i = \text{ZERO}(y^i)$ for each i , and $B = \cup_i B_i$. Use the binary search procedure to compute $h(x)$, and **Reject** if $h(x) = \text{nil}$.

2.1 Let $\alpha = h(x) \in \text{ZERO}(x)$. **Reject** if $\alpha \in B$.

2.2. Uniformly draw a $P \subseteq B$ of size $r - 1$. **Reject** if $f(P \cup \{\alpha\}) = 1$.

End of Stage 2. Accept.

Figure 2: The distribution-free algorithm for testing monotone conjunctions.

Observation 3.2. *The number of queries used by Algorithm 2 is $O((n^{1/3}/\epsilon^5) \cdot \log^7(n/\epsilon))$.*

Observation 3.3. *All queries to the sampling oracle are made in Stage 0.*

Next we prove that this is indeed a one-sided algorithm for testing monotone conjunctions.

Lemma 3.4. *If $f \in \text{MCONJ}$, then Algorithm 2 always accepts (f, \mathcal{D}) for any distribution \mathcal{D} over $\{0, 1\}^n$.*

Proof. Since Algorithm 2 always accepts at the end of Stage 2, it suffices to show that it never rejects when f is a monotone conjunction. First note that $f(1^n)$ must be 1 when f is a monotone conjunction. By Property 3.1, $h(x) = \text{nil}$ can never happen in Stage 0 when f is a monotone conjunction and $x \in f^{-1}(0)$.

This leaves us to check lines 1.1, 1.2, 2.1 and 2.2. Assume that $f \in \text{MCONJ}$:

1. If $B_1, \dots, B_k \subseteq [n]$ satisfy $f(B_1) = \dots = f(B_k) = 1$, then every $Z \subseteq \cup_i B_i$ satisfies $f(Z) = 1$.

This implies that Algorithm 2 never rejects on line 1.1, 1.2 or 2.1.

2. For line 2.2, $\alpha = h(x)$ implies that $f(\{\alpha\}) = 0$ which implies that $f(P \cup \{\alpha\}) = 0$ when f is a monotone conjunction. So Algorithm 2 never rejects on line 2.2.

This finishes the proof of the lemma. □

Theorem 1.1 follows directly from the following lemma combined with Observation 3.2 and Lemma 3.4 (since Algorithm 2 is one-sided its success probability in Lemma 3.5 can be easily amplified to $2/3$).

Lemma 3.5. *If f is ϵ -far from MCONJ with respect to \mathcal{D} , Algorithm 2 rejects with probability at least 0.1.*

3.3 Reduction to Well-Supported Probability Distributions

To ease the proof of Lemma 3.5, we show that it suffices to focus on so-called well-supported distributions. We say a probability distribution \mathcal{D} on $\{0, 1\}^n$ is *well-supported* with respect to f if every empty string of f has probability zero in \mathcal{D} . Given f , intuitively an adversary to pair it with a hard probability distribution \mathcal{D} may not want to allocate much probability on empty points of f , in case Algorithm 2 rejects in Stage 0.

Following this intuition that well-supported probability distributions are probably hard cases of Lemma 3.5, we prove Lemma 3.6 below concerning such distributions in the rest of the section. Before its proof we show that it indeed implies Lemma 3.5.

Lemma 3.6. *Assume that f is a Boolean function and \mathcal{D}' is a well-supported distribution with respect to f . If f is $(\epsilon/2)$ -far from MCONJ with respect to \mathcal{D}' , Algorithm 2 rejects (f, \mathcal{D}') with probability at least 0.1.*

Proof of Lemma 3.5 assuming Lemma 3.6. Assume that f is ϵ -far from MCONJ with respect to \mathcal{D} . Let $\delta \geq 0$ denote the total probability of \mathcal{D} over empty strings of f . If $\delta = 0$, Lemma 3.5 follows directly from Lemma 3.6 since \mathcal{D} is well-supported. If $\delta \geq \epsilon/2$, Algorithm 2 must reject with probability $1 - o(1)$ in Stage 0. We consider below the remaining case when $0 < \delta < \epsilon/2$.

Let \mathcal{D}' denote the following distribution derived from \mathcal{D} . The probability of any empty string of f in \mathcal{D}' is 0. The probability of any other string is set to be its probability in \mathcal{D} multiplied by $1/(1 - \delta)$. Clearly \mathcal{D}' is now a well-supported probability distribution with respect to f . We prove the following claim:

Claim 3.7. *The probability of Algorithm 2 rejecting (f, \mathcal{D}) is at least as large as that of rejecting (f, \mathcal{D}') .*

Proof. Algorithm 2 always rejects (f, \mathcal{D}) if one of the samples in Stage 0 is an empty string. Let E denote the event that no sample in Stage 0 is empty. Then the probability of Algorithm 2 accepting (f, \mathcal{D}') is exactly that of it accepting (f, \mathcal{D}) conditioning on E . This follows from the definition of \mathcal{D}' and our observation 3.3: Stages 1 and 2 access the black-box oracle only, which does not involve \mathcal{D} or \mathcal{D}' . As a result, we have

$$\Pr[(f, \mathcal{D}) \text{ accepted}] = \Pr[(f, \mathcal{D}) \text{ accepted} \mid E] \cdot \Pr[E] \leq \Pr[(f, \mathcal{D}') \text{ accepted}].$$

This finishes the proof of the claim. □

Finally we show that f is $(\epsilon/2)$ -far from MCONJ with respect to \mathcal{D}' . Given this we can then apply Claim 3.7 to finish the proof of the lemma. To see this is the case, note that the total variation distance $d_{TV}(\mathcal{D}, \mathcal{D}')$ is δ by the definition of \mathcal{D}' . This implies that for any Boolean function g , we have

$$|\text{dist}_{\mathcal{D}}(f, g) - \text{dist}_{\mathcal{D}'}(f, g)| \leq d_{TV}(\mathcal{D}, \mathcal{D}') \leq \delta.$$

As a result, $\text{dist}_{\mathcal{D}'}(f, \text{MCONJ}) \geq \text{dist}_{\mathcal{D}}(f, \text{MCONJ}) - \delta \geq \epsilon/2$. This finishes the proof of the lemma. □

We prove Lemma 3.6 in the rest of the section. For convenience, we still use \mathcal{D} to denote the unknown distribution, but from now on we always assume without loss of generality that 1) \mathcal{D} is well-supported with respect to f , and 2) f is $(\epsilon/2)$ -far from MCONJ with respect to \mathcal{D} .

It is worth mentioning that since \mathcal{D} is well-supported, Algorithm 2 can skip Stage 0, which is the reason why it is named Stage 0, and have both Stage 1 and each iteration of Stage 2 start by making $3t$ new queries to the sampling oracle. We will follow this view in the analysis of Algorithm 2 in the rest of the section.

3.4 The Violation Bipartite Graph

We first review the *violation hypergraph* of a Boolean function f introduced by Dolev and Ron [DR11]. It inspires us to define the *violation bipartite graph* G_f of f . The latter is conceptually simpler, and characterizes the distance of f to MCONJ as well. The main lemma of this subsection shows that if $\text{dist}_{\mathcal{D}}(f) \geq \epsilon/2$, then G_f has a *highly regular* subgraph G_f^* with vertex covers of weight $\Omega(\epsilon)$ only.

We start with the definition of the violation hypergraph of a given $f : \{0, 1\}^n \rightarrow \{0, 1\}$ from [DR11].

Definition 3.8 (Violation Hypergraph). *Given f , we call $H_f = (V(H_f), E(H_f))$ the violation hypergraph of f , where $V(H_f) = \{0, 1\}^n$; $E(H_f)$ contains all subsets $\{x, y^1, \dots, y^t\} \subseteq \{0, 1\}^n$ such that*

- $f(x) = 0$; $f(y^i) = 1$ for all $i : 1 \leq i \leq t$; and $\text{ZERO}(x) \subseteq \cup_{i=1}^t \text{ZERO}(y^i)$.

Note that $\{1^n\} \in E(H_f)$ if $f(1^n) = 0$ (this is the only possible special case when $t = 0$).

It turns out that $\text{dist}_{\mathcal{D}}(f, \text{MCONJ})$ is characterized by weights of vertex covers of H_f .

Lemma 3.9 (Lemmas 3.2 and 3.4 of [DR11]). *A function f is in MCONJ if and only if $E(H_f) = \emptyset$. For any Boolean function f , every vertex cover C of H_f has total probability $\mathcal{D}(C) \geq \text{dist}_{\mathcal{D}}(f, \text{MCONJ})$.*

Note that this lemma holds for any (not necessarily well-supported) probability distribution \mathcal{D} . Now we define the violation bipartite graph of f .

Definition 3.10 (Violation Bipartite Graph). *Given a Boolean function f we call the following graph $G_f = (L \cup R, E)$ the violation bipartite graph of f : vertices on the left side are $L = f^{-1}(1)$; vertices on the right side are $R = \{j \in [n] : x \in f^{-1}(0) \text{ and } h(x) = j\}$; add an edge between $y \in f^{-1}(1)$ and $j \in R$ if $y_j = 0$.*

Let \mathcal{D} be a probability distribution over $\{0, 1\}^n$. It defines a nonnegative weight $\text{wt}_{\mathcal{D}}(\cdot)$ for each vertex in G_f as follows. The weight of $y \in f^{-1}(1) = L$ is simply $\text{wt}_{\mathcal{D}}(y) = \mathcal{D}(y)$. The weight of $j \in R$ is

$$\text{wt}_{\mathcal{D}}(j) = \sum_{x \in f^{-1}(0) : h(x)=j} \mathcal{D}(x).$$

Given a set of vertices $C \subseteq L \cup R$, we let $\text{wt}_{\mathcal{D}}(C)$ denote the total weight of C : $\text{wt}_{\mathcal{D}}(C) = \sum_{u \in C} \text{wt}_{\mathcal{D}}(u)$. Most of the time when \mathcal{D} is clear from the context, we drop the subscript and use simply wt for the weight.

From now on we assume that \mathcal{D} is well-supported with respect to f . We get the following corollary:

Corollary 3.11. *If \mathcal{D} is well-supported, then every vertex cover C of G_f has $\text{wt}(C) \geq \text{dist}_{\mathcal{D}}(f, \text{MCONJ})$.*

Proof. Given a vertex cover C of G_f , we define a vertex cover C' of H_f as follows. C' consists of 1) all the empty strings of f ; 2) $C \cap L = C \cap f^{-1}(1)$; and 3) $x \in f^{-1}(0)$ such that $h(x) \neq \text{nil}$ and $h(x) \in C \cap R$.

By the definition of C' and $\text{wt}(\cdot)$, we have $\text{wt}(C') = \mathcal{D}(C')$ (\mathcal{D} is well-supported so has zero probability on empty strings). It suffices to show that C' is a vertex cover of H_f , and then apply Lemma 3.9.

Fix a hyperedge $\{x, y^1, \dots, y^t\}$ in H_f . For the special case when $t = 0$, we have $x = 1^n$ and $f(1^n) = 0$. So 1^n is empty, and $1^n \in C'$. When $t \geq 1$, either $h(x) = \text{nil}$, for which case we have $x \in C'$, or $h(x) \neq \text{nil}$ and $h(x) \in \text{ZERO}(x)$. The latter implies $h(x) \in \text{ZERO}(y^k)$, for some $k \in [t]$, and thus, $(y^k, h(x))$ is an edge in G_f . Since C covers this edge, either $y^k \in C'$ or $x \in C'$. This finishes the proof of the lemma. \square

Next, we extract from G_f a highly regular bipartite graph G_f^* , with the guarantee that any vertex cover of G_f^* still has total weight $\Omega(\epsilon)$ (recall that $\text{dist}_{\mathcal{D}}(f, \text{MCONJ}) \geq \epsilon/2$). We start with some notation. Given a subgraph $G = (L(G) \cup R(G), E(G))$ induced by $L(G) \subseteq L$ and $R(G) \subseteq R$, the *weight* of graph G is

$$\text{wt}(G) = \sum_{y \in L(G)} \text{wt}(y) \cdot \deg_G(y).$$

where $\deg_G(y)$ is the degree of y in G . Equivalently, one can assign each edge (y, j) in G_f an edge weight of $\text{wt}(y)$, and $\text{wt}(G)$ is its total edge weight. For each $j \in R(G)$, we define its *incoming weight* as

$$\text{in-wt}(j) = \sum_{y: (y,j) \in E(G)} \text{wt}(y),$$

which can be viewed as the total edge weight from edges incident to j .

Recall the parameter d in (1). We say a vertex $y \in L(G)$ is *heavy* in G if $\deg_G(y) \geq d \cdot \text{wt}(G)$; a vertex $j \in R(G)$ is *heavy* in G if $\text{in-wt}(j) \geq d \cdot \text{wt}(G) \cdot \text{wt}(j)$. In either cases, removing a heavy vertex u (and its incident edges) would reduce $\text{wt}(G)$ by $\geq d \cdot \text{wt}(G) \cdot \text{wt}(u)$. We say a vertex is *light* if it is not heavy.

We run the following deterministic procedure on G_f to define a subgraph G_f^* of G_f . (This procedure is not new and has seen many applications in the literature, e.g., see [RM99].)

1. Let $G = G_f$ and $S = \emptyset$. Remove all vertices in G with degree zero.
2. Remove all heavy vertices on the left side of G and their incident edges, if any; move them to S . Also remove vertices on the right side that now have degree zero.
3. If G has a vertex cover C of total (vertex) weight $\text{wt}(C) \leq \epsilon/4$, exit.
4. Remove all heavy vertices on the right side of G and their incident edges, if any; move them to S . Also remove vertices on the left side that now have degree zero.
5. If G has a vertex cover C of total (vertex) weight $\text{wt}(C) \leq \epsilon/4$ or there exists no more heavy vertex in G , exit; otherwise go back to Step 2.

Let $G_f^* = (L^* \cup R^*, E^*)$ denote the subgraph of G_f induced by $L^* \subseteq L$ and $R^* \subseteq R$ we obtain at the end.

We show that G_f^* has no heavy vertex, and any vertex cover C of G_f^* still has a large total weight.

Lemma 3.12. *Assume that \mathcal{D} is well-supported with respect to f and they satisfy $\text{dist}_{\mathcal{D}}(f, \text{MCONJ}) \geq \epsilon/2$. Then G_f^* has no heavy vertex, and any of its vertex cover C has a total weight of $\text{wt}(C) \geq 3\epsilon/8$.*

Proof. The first part, i.e. G_f^* has no heavy vertex, follows from the second part of the lemma, which implies that the procedure exits because G contains no more heavy vertex.

The second part follows from the claim that $\text{wt}(S) = o(\epsilon)$ (as for any vertex cover C of G_f^* , $C \cup S$ is a vertex cover of G_f but by Corollary 3.11, $\text{wt}(C \cup S) \geq \epsilon/2$). To prove the claim, we let G_0, \dots, G_s denote the sequence of graphs obtained by following the procedure, with $G_f = G_0$ and $G_s = G_f^*$, and let S_i denote the set of vertices that are removed from G_i to obtain G_{i+1} and added to S . (Note that S_i does not include those vertices removed because their degrees drop to zero.) By the definition of heavy vertices, we have

$$\text{wt}(G_i) - \text{wt}(G_{i+1}) \geq d \cdot \text{wt}(G_i) \cdot \text{wt}(S_i).$$

Given this connection, we upperbound $\text{wt}(S) = \sum_{i=0}^{s-1} \text{wt}(S_i)$ by analyzing the following sum:

$$\sum_{i=0}^{s-1} \frac{\text{wt}(G_i) - \text{wt}(G_{i+1})}{\text{wt}(G_i)} \leq 1 + \sum_{i=0}^{s-2} \int_{\text{wt}(G_{i+1})}^{\text{wt}(G_i)} (1/u) du = 1 + \int_{\text{wt}(G_0)}^{\text{wt}(G_{s-1})} (1/u) du = O(\log(n/\epsilon)),$$

where the last inequality follows from $\text{wt}(G_0) \leq n$ and $\text{wt}(G_{s-1}) \geq \epsilon/4$ (since any of its vertex cover, e.g., by taking all vertices on the left side, has weight at least $\epsilon/4$). Thus,

$$\text{wt}(S) = \sum_{i=0}^{s-1} \text{wt}(S_i) \leq \frac{1}{d} \cdot \sum_{i=0}^{s-1} \frac{\text{wt}(G_i) - \text{wt}(G_{i+1})}{\text{wt}(G_i)} = o(\epsilon),$$

by the choice of d in (1). This finishes the proof of the lemma. \square

Note that because any of its vertex cover has weight $\Omega(\epsilon)$, we have $\text{wt}(L^*) = \Omega(\epsilon)$. Let $W = \text{wt}(G_f^*)$. Then we also have $W = \Omega(\epsilon)$ simply because every vertex in G_f^* has degree at least one. Since all vertices are light, we have in G_f^* that $\deg(y) \leq d \cdot W$ for all $y \in L^*$ and $\text{in-wt}(j) \leq d \cdot W \cdot \text{wt}(j)$ for all $j \in R^*$.

The bipartite graph G_f^* is extremely useful for the analysis of our algorithm later. Before that we make a short detour to sketch an informal analysis of the tester of Dolev and Ron [DR11] (note that our dependency on ϵ here is worse than their analysis) which may help the reader better understand the construction so far.

First, let $R' \subseteq R^*$ be the set of vertices $j \in R^*$ such that $\text{in-wt}(j) \geq \text{wt}(j) \cdot W/2$. Then

$$W = \sum_{j \in R^*} \text{in-wt}(j) \leq (W/2) \cdot \sum_{j \notin R'} \text{wt}(j) + d \cdot W \cdot \sum_{j \in R'} \text{wt}(j) \leq (W/2) + d \cdot W \cdot \text{wt}(R'),$$

which implies that $\text{wt}(R') = \Omega(1/d)$. Moreover, every $S \subseteq R'$ satisfies the following nice property (below we use $N(S)$ to denote the set of neighbors of S in G_f^*):

Lemma 3.13. *In G_f^* , every $S \subseteq R'$ satisfies $\text{wt}(N(S)) = \Omega(\text{wt}(S)/d)$.*

Proof. Let $W = \text{wt}(G_f^*)$. The total edge weight between S and $N(S)$ in G_f^* is

$$\sum_{j \in S} \text{in-wt}(j) \leq \sum_{y \in N(S)} \deg(y) \cdot \text{wt}(y).$$

Because $S \subseteq R'$, the LHS is at least

$$\sum_{j \in S} \text{in-wt}(j) \geq (W/2) \cdot \sum_{j \in S} \text{wt}(j) = (W/2) \cdot \text{wt}(S).$$

Since there is no heavy vertex in G_f^* , the RHS is at most

$$\sum_{y \in N(S)} \deg(y) \cdot \text{wt}(y) \leq d \cdot W \cdot \sum_{y \in N(S)} \text{wt}(y) = d \cdot W \cdot \text{wt}(N(S)).$$

The lemma follows by combining all these inequalities. \square

Remark 3.14. *We use G_f^* and R' to sketch an alternative and informal analysis of the tester of Dolev and Ron [DR11] for well-supported distributions \mathcal{D} (which can be extended to general distributions). Below we assume that ϵ is a constant for convenience so the dependency on ϵ is worse than that of [DR11]. The tester starts by sampling a set T of $\tilde{O}(\sqrt{n})$ pairs from the sampling oracle. It then claims victory if there are two strings x and y from T such that $f(x) = 0$, $f(y) = 1$, and $(y, h(x))$ is an edge in G_f .*

Let T_1 denote the set of 1-strings, and T_0 denote the set of 0-strings from T . Also let $R'' \subseteq R'$ denote the set of $j \in R'$ such that $h(x) = j$ for some $x \in T_0$. Since $\mathcal{D}(R') = \text{wt}(R') = \tilde{\Omega}(1)$, we have $\text{wt}(R'') = \tilde{\Omega}(1/\sqrt{n})$ with high probability (here $\tilde{O}(\sqrt{n})$ samples suffice because there are only n coordinates). When this happens, by Lemma 3.13 we have $\text{wt}(N(R'')) = \tilde{\Omega}(1/\sqrt{n})$ as well. The tester then rejects if one of the samples in T_1 lies in $N(R'')$. This should happen with high probability if we set the hidden polylogarithmic factor in the number of queries large enough.

Now we return to the analysis of our algorithm (actually we will not use R' in our analysis). Recall that $W = \text{wt}(G_f^*)$. Let $L' \subseteq L^*$ denote the set of $y \in L^*$ such that $\deg(y) \geq W/2$ in G_f^* . Then similarly

$$W = \sum_{y \in L^*} \deg(y) \cdot \text{wt}(y) \leq (W/2) \cdot \sum_{y \notin L'} \text{wt}(y) + d \cdot W \cdot \sum_{y \in L'} \text{wt}(y) \leq (W/2) + d \cdot W \cdot \text{wt}(L'),$$

which implies that $\text{wt}(L') \geq 1/(2d)$. Our analysis of Algorithm 2 heavily relies on G_f^* and $L' \subseteq L^*$.

We summarize below all the properties we need about G_f^* and L' .

Property 3.15. *Assume that \mathcal{D} is well-supported with respect to f and $\text{dist}_{\mathcal{D}}(f, \text{MCONJ}) \geq \epsilon/2$. Then $G_f^* = (L^* \cup R^*, E^*)$ and $L' \subseteq L^*$ defined above have the following properties (letting $W = \text{wt}(G_f^*)$).*

1. $W = \Omega(\epsilon)$ and $\text{wt}(L') \geq 1/(2d)$.
2. $\text{in-wt}(j) \leq d \cdot W \cdot \text{wt}(j)$ for all $j \in R^*$. (We only use the fact that vertices in R^* are light.)
3. Every $y \in L'$ has $\deg(y) \geq \max(1, W/2)$.

3.5 Analysis of Algorithm 2

We now prove Lemma 3.6. Let \mathcal{D} be a well-supported probability distribution with respect to $f : \{0, 1\}^n \rightarrow \{0, 1\}$, such that f is $(\epsilon/2)$ -far from MCONJ with respect to \mathcal{D} . Let $G_f^* = (L^* \cup R^*, E^*)$ denote the bipartite graph defined using f and \mathcal{D} in the previous subsection, with G_f^* and $L' \subseteq L^*$ satisfying Property 3.15.

Here is a sketch of the proof. We first analyze Stages 1 and 2 of Algorithm 2 in Section 3.5.1, where we show that if a sequence of t samples (y^1, \dots, y^t) passes Stage 1 with high probability then it can be used to produce many sequences of strings that get rejected in Stage 2 with high probability. Then in Section 3.5.2, assuming that (f, \mathcal{D}) passes Stage 1 with high probability without loss of generality, we use G_f^* to show that (f, \mathcal{D}) must get rejected in Stage 2 with high probability, where Property 3.15 plays a crucial role.

3.5.1 Analysis of Stages 1 and 2

First we assume without loss of generality that $f(\mathbf{1}^n) = 1$; otherwise it is rejected at the beginning of Stage 0. As f is $(\epsilon/2)$ -far from MCONJ, we have that both $\mathcal{D}(f^{-1}(0))$ and $\mathcal{D}(f^{-1}(1))$ are at least $\epsilon/2$. The former follows trivially from the fact that the all-1 function is in MCONJ. For the latter, we only need to observe that the distance between f and the conjunction of all n variables with respect to \mathcal{D} is at most $\mathcal{D}(f^{-1}(1))$, given $f(\mathbf{1}^n) = 1$.

Recall that since \mathcal{D} is well-supported with respect to f , we can skip Stage 0 and have Stage 1 and each iteration of Stage 2 start by drawing $(3t/\epsilon)$ fresh samples from the sampling oracle. It follows directly from Chernoff bound that Stage 1 reaches Step 1.1 with probability $1 - o(1)$. Let \mathcal{D}^1 denote the distribution of $y \in_R \mathcal{D}$ conditioning on $y \in f^{-1}(1)$. Equivalently, we have that Stage 1 accepts with probability $o(1)$, and with probability $1 - o(1)$ it draws a sequence of t samples y^1, \dots, y^t independently from \mathcal{D}^1 and then goes through Steps 1.1 and 1.2.

The same can be said about Stage 2: Stage 2 accepts with probability $o(1)$ by Chernoff bound and a union bound; with probability $1 - o(1)$, each iteration of Stage 2 draws a sequence of $t - 1$ samples y^1, \dots, y^{t-1} from \mathcal{D}^1 as well as one sample x from $f^{-1}(0)$, proportional to $\mathcal{D}(x)$. Since Steps 2.1 and 2.2 use only $\alpha = h(x)$ but not the string x itself, this inspires us to introduce \mathcal{D}^0 as the distribution over R proportional to $\text{wt}(j)$, $j \in R$. Hence equivalently, each iteration of Stage 2 draws an index α from \mathcal{D}^0 and goes through Steps 2.1 and 2.2 using y^i and α .

We introduce some notation. Let $\mathcal{B} = (B_1, \dots, B_t)$ be a sequence of t (not necessarily distinct) 1-sets of f (i.e., $f(B_i) = 1$). We refer to \mathcal{B} as a 1-sequence of length t . Let $B = \cup_i B_i$. We say \mathcal{B} passes Stage 1 with probability c if \mathcal{B} passes Steps 1.1 and 1.2 with probability c , without being rejected. Similarly, we let $\mathcal{B} = (B_1, \dots, B_{t-1})$ denote a 1-sequence of length $t - 1$, with $B = \cup_i B_i$. Let $\alpha \in R$. Then we say (\mathcal{B}, α) fails an iteration of Stage 2 with probability c if (\mathcal{B}, α) gets rejected in Steps 2.1 or 2.2 with probability c .

We now analyze 1-sequences $\mathcal{B} = (B_1, \dots, B_t)$ that pass Stage 1 with high probability. Let

$$B_i^* = B_i - \cup_{j \neq i} B_j, \quad \text{for each } i \in [t].$$

So B_i^* contains indices that are unique to B_i among all sets in \mathcal{B} . Let $I_{\mathcal{B}}$ denote the set of $i \in [t]$ such that $y_i \in L'$, where y_i is the 1-string with $\text{ZERO}(y_i) = B_i$. Intuitively, $|I_{\mathcal{B}}|$ should be large with high probability since $\mathcal{D}(L') = \text{wt}(L')$ is large by Property 3.15. We say \mathcal{B} is *strong* if $|I_{\mathcal{B}}| \geq t/(3d) = r/3$. Moreover, let $I_{\mathcal{B}}^*$ denote the set of $i \in I_{\mathcal{B}}$ such that $|B_i^*| \leq 6|B|/r$.

By an averaging argument we show that if \mathcal{B} is strong then $|I_{\mathcal{B}}^*|$ is at least $r/6$.

Lemma 3.16. *If \mathcal{B} is strong, then we have $|I_{\mathcal{B}}^*| \geq r/6$.*

Proof. As $\sum_i |B_i^*| \leq |B|$, the number of B_i with $|B_i^*| > 6|B|/r$ is at most $r/6$. The lemma follows. \square

Let $\mathcal{B} = (B_1, \dots, B_t)$ denote a strong 1-sequence of length t and y_i denote the string with $\text{ZERO}(y_i) = B_i$. We use it to generate input pairs (\mathcal{B}', α) to Stage 2, where \mathcal{B}' is a 1-sequence of length $t-1$ and $\alpha \in R$, as follows. For each pair (i, α) such that $i \in I_{\mathcal{B}}^*$ and $\alpha \in B_i \cap R^*$, we say \mathcal{B} generates (\mathcal{B}', α) via (i, α) if

$$\mathcal{B}' = (B_1, \dots, B_{i-1}, B_{i+1}, \dots, B_t),$$

and we call such (i, α) a *valid* pair. Note that as B_i 's are not necessarily distinct, \mathcal{B} may generate the same pair (\mathcal{B}', α) via (i, α) and (j, α) , $i \neq j$. In the main technical lemma of this section, Lemma 3.19 below, we show that if \mathcal{B} is strong and passes Stage 1 with high probability, then many (i, α) would lead to pairs (\mathcal{B}', α) that fail Stage 2 with high probability. Before that we make a few observations. Recall $W = \text{wt}(G_f^*)$.

Observation 3.17. *Since $y_i \in L'$, we have $B_i \cap R^* = \deg(y_i)$ in G_f^* and $|B_i \cap R^*| \geq \max(1, W/2)$. So the total number of valid pairs (i, α) is bounded from below by both $r/6$ and $rW/12$.*

Observation 3.18. *If a valid pair (i, α) satisfies $\alpha \in B_i \setminus B_i^*$ (i.e., α is shared by another B_j in \mathcal{B}), then it generates a pair (\mathcal{B}', α) that fails Stage 2 (Step 2.1) with probability 1.*

Now we prove Lemma 3.19.

Lemma 3.19. *Assume that $\mathcal{B} = (B_1, \dots, B_t)$ is a strong 1-sequence that passes Stage 1 with probability at least $1/2$. Then there are at least $\Omega(rW)$ many valid (i, α) such that the pair (\mathcal{B}', α) generated by \mathcal{B} via (i, α) fails an iteration of Stage 2 with probability at least $\Omega(1)$ (a constant that does not depend on n or ϵ).*

Proof. For convenience, we use I to denote $I_{\mathcal{B}}^*$, with $|I| = \Omega(r)$ because \mathcal{B} is strong (Lemma 3.16). We let $B^* = \cup_{i \in I} B_i^*$, and let $\Gamma = B^* \cap R^*$ (which can be empty). We first consider two special cases on $|\Gamma|$.

Case 1: $|\Gamma| > |B|/t$. Note that every $j \in \Gamma$ satisfies $f(\{j\}) = 0$. This implies that \mathcal{B} would get rejected with probability $1 - o(1)$ in Step 1.1, contradicting the assumption that \mathcal{B} passes it with probability $1/2$.

Case 2: $|\Gamma| < rW/24$. By Observation 3.17, the number of valid pairs (i, α) is at least $rW/12$. In this case, however, the number of valid pairs (i, α) such that $\alpha \in B_i^*$ is at most $rW/24$. Thus, the number of valid pairs (i, α) such that $\alpha \in B_i \setminus B_i^*$ is at least $rW/24$. The lemma follows from Observation 3.18.

In the rest of the proof we assume that $|B| \geq t|\Gamma|$ and $|\Gamma| = \Omega(rW)$. They together imply that

$$|B| \geq t|\Gamma| = \Omega(rtW). \quad (2)$$

For $\alpha \in \Gamma$ let $s_\alpha \in [t]$ be the unique index with $\alpha \in B_{s_\alpha}^*$. Now we need to do some counting.

Let \mathcal{Z} denote the set of all subsets $Z \subset B$ of size r such that $f(Z) = 1$. Since we assumed that \mathcal{B} passes Stage 1 with probability at least $1/2$, it must be the case that

$$|\mathcal{Z}| \geq \left(1 - O\left(\frac{1}{s}\right)\right) \cdot \binom{|B|}{r}.$$

Fixing an $\alpha \in \Gamma$ with $\alpha \in B_{s_\alpha}^*$, we are interested in

$$\mathcal{S}_\alpha = \left\{ P \cup \{\alpha\} : P \text{ is a subset of } B \setminus B_{s_\alpha}^* \text{ of size } r-1 \right\} \quad \text{and} \quad N_\alpha = |\mathcal{S}_\alpha \cap \mathcal{Z}|.$$

We would like to prove a strong lower bound for $\sum_{\alpha \in \Gamma} N_\alpha$.

To give some intuition on the connection between N_α and the goal, notice that $B \setminus B_{s_\alpha}^* = \cup_{i \neq s_\alpha} B_i$. Let (\mathcal{B}', α) be the pair generated from \mathcal{B} via (s_α, α) . If a set P of size $r-1$ is drawn from $\cup_{i \neq s_\alpha} B_i$ uniformly at random, then the probability of P leading Step 2.2 to reject (\mathcal{B}', α) , denoted by q_α , is

$$q_\alpha = \frac{N_\alpha}{\binom{|B \setminus B_{s_\alpha}^*|}{r-1}} \geq \frac{N_\alpha}{\binom{|B|}{r-1}} = \frac{N_\alpha}{\binom{|B|}{r} \cdot \frac{r}{|B|-r+1}} \geq \frac{N_\alpha}{\binom{|B|}{r}} \cdot \frac{|B|}{2r},$$

where the last inequality used (2) that $|B| \gg r$. So a strong bound for $\sum_{\alpha \in \Gamma} N_\alpha$ may lead us to the desired claim that q_α is large for most $\alpha \in \Gamma$. To bound $\sum_{\alpha \in \Gamma} N_\alpha$ and avoid double counting, let

$$\mathcal{S}'_\alpha = \left\{ P \cup \{\alpha\} : P \text{ is a subset of } B \setminus (B_{s_\alpha}^* \cup \Gamma) \text{ of size } r-1 \right\} \quad \text{and} \quad N'_\alpha = |\mathcal{S}'_\alpha \cap \mathcal{Z}|.$$

Since $\mathcal{S}'_\alpha \subseteq \mathcal{S}_\alpha$ and \mathcal{S}'_α are now pairwise disjoint, we have $\sum_\alpha N_\alpha \geq \sum_\alpha N'_\alpha$ and

$$\sum_{\alpha \in \Gamma} N'_\alpha = \left| \left(\cup_{\alpha \in \Gamma} \mathcal{S}'_\alpha \right) \cap \mathcal{Z} \right| \geq \left| \cup_{\alpha \in \Gamma} \mathcal{S}'_\alpha \right| + |\mathcal{Z}| - \binom{|B|}{r} \geq \sum_{\alpha \in \Gamma} |\mathcal{S}'_\alpha| - O\left(\frac{1}{s}\right) \cdot \binom{|B|}{r}.$$

On the other hand, by the definition of I_B^* we have $|B_{s_\alpha}^*| \leq 6|B|/r$. We also have $|\Gamma| \leq |B|/t$. Thus

$$|\mathcal{S}'_\alpha| = \binom{|B \setminus (B_{s_\alpha}^* \cup \Gamma)|}{r-1} \geq \binom{|B| - (7|B|/r)}{r-1} = \Omega\left(\frac{r}{|B|} \cdot \binom{|B|}{r}\right), \quad (3)$$

where details of the last inequality can be found in Appendix A.

Using $|\Gamma| = \Omega(rW)$ and $W = \Omega(\epsilon)$, $r = n^{1/3}$ and $|B| \leq n$, we have

$$\sum_{\alpha \in \Gamma} |\mathcal{S}'_\alpha| = \Omega\left(\frac{r|\Gamma|}{|B|} \cdot \binom{|B|}{r}\right) = \omega\left(\left(\frac{1}{s}\right) \cdot \binom{|B|}{r}\right).$$

As a result, we obtain the following lower bound for $\sum_{\alpha \in \Gamma} N_\alpha$:

$$\sum_{\alpha \in \Gamma} N_\alpha = \Omega\left(\frac{r|\Gamma|}{|B|} \cdot \binom{|B|}{r}\right).$$

Combining the connection between N_α and q_α , we have $\sum_{\alpha \in \Gamma} q_\alpha = \Omega(|\Gamma|)$. Since $q_\alpha \leq 1$ (it is a probability) for all α , it follows easily that $q_\alpha = \Omega(1)$ for $\Omega(|\Gamma|)$ many α 's in Γ . For each such α , (s_α, α) is a valid pair via which \mathcal{B} generates a pair (\mathcal{B}', α) that gets rejected by Stage 2 with probability $\Omega(1)$.

The lemma then follows from $|\Gamma| = \Omega(rW)$. \square

3.5.2 Finishing the Proof of Lemma 3.6

Now we combine Lemma 3.19 and G_f^* , L' to finish the proof of Lemma 3.6.

Assume without loss of generality that Stage 1 of Algorithm 2 either accepts (f, \mathcal{D}) or passes it down to Stage 2 with probability at least 0.9; otherwise we are already done.

Recall that \mathcal{D}^1 is the distribution of $y \in_R \mathcal{D}$ conditioning on $y \in f^{-1}(1)$. We abuse the notation a little

bit and also use \mathcal{D}^1 to denote the corresponding distribution on 1-sets. Given a 1-sequence $\mathcal{B} = (B_1, \dots, B_t)$ of length t , we write $p(\mathcal{B}) = \Pr_{\mathcal{D}^1}[B_1] \times \dots \times \Pr_{\mathcal{D}^1}[B_t]$. From our discussion earlier, Stage 1 accepts (f, \mathcal{D}) with probability $o(1)$, and with probability $1 - o(1)$, it runs Steps 1.1 and 1.2 on a 1-sequence \mathcal{B} with each entry B_i drawn from \mathcal{D}^1 independently. This implies that

$$\sum_{\text{1-seq } \mathcal{B}} p(\mathcal{B}) \cdot \Pr[\mathcal{B} \text{ passes Stage 1}] \geq 0.8.$$

We focus on strong 1-sequences. We write S to denote the set of strong 1-sequences and let S' denote the set of strong 1-sequences that pass Stage 1 with probability at least $1/2$. Because $\mathcal{D}(L') = \text{wt}(L') \geq 1/(2d)$ we have that Stage 1 draws a strong \mathcal{B} with probability $1 - o(1)$ by Chernoff bound. As a result, we have

$$\sum_{\mathcal{B} \in S} p(\mathcal{B}) \cdot \Pr[\mathcal{B} \text{ passes Stage 1}] \geq 0.8 - o(1) > 0.7.$$

But the LHS is at most

$$\sum_{\mathcal{B} \in S} p(\mathcal{B}) \cdot \Pr[\mathcal{B} \text{ passes Stage 1}] \leq (1/2) \cdot \sum_{\mathcal{B} \in S \setminus S'} p(\mathcal{B}) + \sum_{\mathcal{B} \in S'} p(\mathcal{B}) \leq (1/2) + \sum_{\mathcal{B} \in S'} p(\mathcal{B})$$

and thus, $\sum_{\mathcal{B} \in S'} p(\mathcal{B}) = \Omega(1)$. The remaining proof is to use this (combined with Lemma 3.19, G_f^* and L') to show that a random pair (\mathcal{B}', α) gets rejected in Stage 2 with high probability.

To this end, recall that \mathcal{D}^0 is the distribution over R proportional to $\text{wt}(j)$, $j \in R$. For each pair (\mathcal{B}', α) , where \mathcal{B}' is a 1-sequence of length $t-1$ and $\alpha \in R$, let $q(\mathcal{B}', \alpha) = \Pr_{\mathcal{D}^1}[B'_1] \times \dots \times \Pr_{\mathcal{D}^1}[B'_{t-1}] \cdot \Pr_{\mathcal{D}^0}[\alpha]$.

Since Stage 2 consists of $d^* = d^2/\epsilon$ iterations, it suffices to show that

$$\sum_{(\mathcal{B}', \alpha)} q(\mathcal{B}', \alpha) \cdot \Pr[(\mathcal{B}', \alpha) \text{ fails an iteration of Stage 2}] = \Omega(\epsilon/d^2), \quad (4)$$

as Stage 2 either accepts with probability $o(1)$, or with probability $1 - o(1)$ each iteration of Stage 2 draws (\mathcal{B}', α) according to $q(\cdot)$ and runs it through Steps 2.1 and 2.2.

To take advantage of Lemma 3.19 we use T to denote the set of (\mathcal{B}', α) that is generated by a \mathcal{B} from S' via a pair (i, α) and fails an iteration of Stage 2 with probability $\Omega(1)$ (the same constant hidden in Lemma 3.19). For (4) it then suffices to show that

$$\sum_{(\mathcal{B}', \alpha) \in T} q(\mathcal{B}', \alpha) = \Omega(\epsilon/d^2). \quad (5)$$

Lemma 3.19 implies that for each \mathcal{B} in S' , there exist $\Omega(rW)$ many valid (i, α) such that the pair generated by \mathcal{B} via (i, α) belongs to T (though these (\mathcal{B}', α) 's are not necessarily distinct). We use $J_{\mathcal{B}}$ to denote these pairs of \mathcal{B} . We also write (\mathcal{B}^i, α) to denote the pair generated by \mathcal{B} via (i, α) for convenience.

Then there is the following connection between probabilities $p(\mathcal{B})$ and $q(\mathcal{B}^i, \alpha)$:

$$q(\mathcal{B}^i, \alpha) = \frac{p(\mathcal{B})}{\Pr_{\mathcal{D}^1}[B_i]} \cdot \Pr_{\mathcal{D}^0}[\alpha] = p(\mathcal{B}) \cdot \frac{\mathcal{D}(f^{-1}(1))}{\mathcal{D}(B_i)} \cdot \frac{\text{wt}(\alpha)}{\text{wt}(R)} \geq \frac{\epsilon}{2} \cdot p(\mathcal{B}) \cdot \frac{\text{wt}(\alpha)}{\mathcal{D}(B_i)},$$

where the inequality follows from $\text{wt}(R) \leq 1$ and $\mathcal{D}(f^{-1}(1)) \geq \epsilon/2$ since f is $(\epsilon/2)$ -far from MCONJ with respect to \mathcal{D} . The only obstacle for (5) is to handle the double counting. This is where G_f^* and L' help.

Consider the following sum (and its connection to (5)):

$$\sum_{\mathcal{B} \in S'} p(\mathcal{B}) \cdot |J_{\mathcal{B}}|. \quad (6)$$

On the one hand, as $|J_{\mathcal{B}}| = \Omega(rW)$ and $\sum_{\mathcal{B} \in S'} p(\mathcal{B}) = \Omega(1)$, the sum is $\Omega(rW)$. On the other hand,

$$(6) = \sum_{\mathcal{B} \in S'} \sum_{(i, \alpha) \in J_{\mathcal{B}}} p(\mathcal{B}) \leq \frac{2}{\epsilon} \cdot \sum_{\mathcal{B} \in S'} \sum_{(i, \alpha) \in J_{\mathcal{B}}} q(\mathcal{B}^i, \alpha) \cdot \frac{\mathcal{D}(B_i)}{\text{wt}(\alpha)}. \quad (7)$$

Focusing on any fixed pair (\mathcal{B}', α) in T , the coefficient of $q(\mathcal{B}', \alpha)$ in (7) is given by

$$\frac{2}{\epsilon \cdot \text{wt}(\alpha)} \cdot \sum_{\substack{\mathcal{B} \in S', (i, \alpha) \in J_{\mathcal{B}} \\ \mathcal{B}^i = \mathcal{B}'}} \mathcal{D}(B_i). \quad (8)$$

However, fixing an $i \in [t]$, for \mathcal{B} to generate (\mathcal{B}', α) via (i, α) , a necessary condition is $\alpha \in B_i$. This implies that the string y satisfying $\text{ZERO}(y) = B_i$ must be a neighbor of α in G_f^* (since $y \in L'$ by definition). As a result it follows from Property 3.15 that the sum of (8) with i fixed is at most $2dW/\epsilon$ (with $\text{wt}(\alpha)$ cancelled) and thus, the coefficient of $q(\mathcal{B}', \alpha)$ of each $(\mathcal{B}', \alpha) \in T$ in (7) is $O(tdW/\epsilon)$.

Combining all these inequalities, we have

$$\Omega(rW) = \sum_{\mathcal{B} \in S'} p(\mathcal{B}) \cdot |J_{\mathcal{B}}| \leq O\left(\frac{tdW}{\epsilon}\right) \cdot \sum_{(\mathcal{B}', \alpha) \in T} q(\mathcal{B}', \alpha),$$

and (4) follows. This finishes the proof of Lemma 3.6, and completes the analysis of Algorithm 2.

4 Lower Bound: Proof of Theorem 1.2

In this section, we present a lower bound of $\tilde{\Omega}(n^{1/3})$ for the distribution-free testing of monotone conjunctions, and prove Theorem 1.2. Our proof is based on techniques used in the $\tilde{\Omega}(n^{1/5})$ lower bound of Glasner and Servedio [GS09], with certain careful modifications on their construction and arguments.

We start by presenting two distributions of pairs (f, \mathcal{D}) , \mathcal{YES} and \mathcal{NO} , in Section 4.1, such that

1. Every pair (f, \mathcal{D}_f) in the support of \mathcal{YES} has $f \in \text{MCONJ}$; and
2. Every pair (g, \mathcal{D}_g) in the support of \mathcal{NO} has $\text{dist}_{\mathcal{D}_g}(g, \text{MCONJ}) \geq 1/3$.

Let $q = n^{1/3}/\log^3 n$. Let T be a deterministic (and adaptive) oracle algorithm that, upon (f, \mathcal{D}) , makes no more than q queries to the sampling oracle and the black-box oracle each. (Note that even though T is deterministic, each of its query to the sampling oracle returns a pair $(x, f(x))$ with x drawn from \mathcal{D} .)

Our main technical lemma in this section shows that T cannot distinguish \mathcal{YES} and \mathcal{NO} .

Lemma 4.1. *Let T be a deterministic oracle algorithm that makes at most q queries to each oracle. Then*

$$\left| \Pr_{(f, \mathcal{D}_f) \sim \mathcal{YES}} [T(f, \mathcal{D}_f) \text{ accepts}] - \Pr_{(g, \mathcal{D}_g) \sim \mathcal{NO}} [T(g, \mathcal{D}_g) \text{ accepts}] \right| \leq \frac{1}{4}.$$

Theorem 1.2 then follows directly from Lemma 4.1 by Yao's minimax lemma.

4.1 The Two Distributions \mathcal{YES} and \mathcal{NO}

We need some notation. For strings $x, y \in \{0, 1\}^n$, we use $x \wedge y \in \{0, 1\}^n$ to denote the bitwise AND of x and y , and $x \vee y \in \{0, 1\}^n$ to denote the bitwise OR of x and y .

We use the following parameters in the definition of the two distributions:

$$h = \frac{n^{2/3}}{2 \log^2 n}, \quad r = n^{1/3} \log^2 n, \quad \ell = n^{2/3} + 2, \quad m = n^{2/3}, \quad \text{and} \quad s = \log^2 n.$$

4.1.1 The Distribution \mathcal{YES}

A draw (f, \mathcal{D}_f) from the distribution \mathcal{YES} is obtained using the following procedure:

1. Select a set R of size $hr + 2m = (n/2) + 2n^{2/3}$ from $[n]$ uniformly at random.
2. Select a tuple of $2m$ different indices $(\alpha_1, \dots, \alpha_m, \beta_1, \dots, \beta_m)$ from R uniformly at random.
3. Partition $R' = R \setminus \{\alpha_1, \dots, \alpha_m, \beta_1, \dots, \beta_m\}$ into r sets of the same size h uniformly at random. We refer to each such set as a *block*.
4. For each $i \in [m]$, select $2 \log^2 n$ blocks uniformly at random (and independently for different i 's) and let C'_i be their union. So $|C'_i| = \ell - 2$. Let $C_i = C'_i \cup \{\alpha_i, \beta_i\}$ for each $i \in [m]$ and thus, $|C_i| = \ell$.
5. For each $i \in [m]$, select $\log^2 n$ blocks from C'_i uniformly at random and call their union together with $\{\alpha_i\}$ to be A_i ; let $B_i = C_i \setminus A_i$. Then A_i and B_i partition C_i and $|A_i| = |B_i| = \ell/2$.
6. We define two Boolean functions $f_1, f_2 : \{0, 1\}^n \rightarrow \{0, 1\}$ as follows:

$$f_1(x_1, \dots, x_n) = \bigwedge_{j \notin R} x_j \quad \text{and} \quad f_2(x_1, \dots, x_n) = x_{\alpha_1} \wedge x_{\alpha_2} \wedge \dots \wedge x_{\alpha_m}.$$

Finally, we define $f : \{0, 1\}^n \rightarrow \{0, 1\}$ as $f(x) = f_1(x) \wedge f_2(x)$.

7. We define distribution \mathcal{D}_g as follows. For each $i \in [m]$, let $a^i, b^i, c^i \in \{0, 1\}^n$ denote the three strings with $A_i = \text{ZERO}(a^i)$, $B_i = \text{ZERO}(b^i)$, and $C_i = \text{ZERO}(c^i)$. Then we have $f(b^i) = 1$ and $f(a^i) = f(c^i) = 0$. The probabilities of b^i and c^i in \mathcal{D}_g are $2/(3m)$ and $1/(3m)$, respectively, for each $i \in [m]$. All other strings have probability zero in \mathcal{D}_g .

It is clear that any pair (f, \mathcal{D}_f) drawn from \mathcal{YES} has $f \in \text{MCONJ}$ as promised earlier.

4.1.2 The Distribution \mathcal{NO}

A draw (g, \mathcal{D}_g) from the distribution \mathcal{NO} is obtained using the following procedure:

1. Follow the first six steps of the procedure for \mathcal{YES} to obtain $R, A_i, B_i, C_i, \alpha_i, \beta_i, f_1, f_2$.
2. We say a string $x \in \{0, 1\}^n$ is *i-special*, for some $i \in [m]$, if it satisfies both conditions:
 - (a) there are at least $3 \log^2 n / 4$ many blocks in A_i , each of which has (strictly) more than s indices j in it with $x_j = 0$; and
 - (b) there are at least $3 \log^2 n / 4$ many blocks in B_i , each of which has at most s indices j in it with $x_j = 0$.
3. We use f_2 to define a new Boolean function g' . If $f_2(x) = 0$ but x is *i-special* for every i such that $x_{\alpha_i} = 0$, then set $g'(x) = 1$; otherwise $g'(x) = f_2(x)$. Finally, we define $g(x) = f_1(x) \wedge g'(x)$.

4. Recall the definition of strings a^i, b^i and c^i from A_i, B_i and C_i . The probability of each of these $3m$ strings a^i, b^i, c^i is set to be $1/(3m)$ in \mathcal{D}_f , and all other strings have probability zero in \mathcal{D}_f .

It's easy to verify that for each (g, \mathcal{D}_g) drawn from the \mathcal{NO} distribution, we have

$$g(a^i) = g(b^i) = 1 \quad \text{but} \quad g(c^i) = g(a^i \wedge b^i) = 0.$$

Note that $f \in \text{MCONJ}$ satisfies $f(x \wedge y) = f(x) \wedge f(y)$. As a result, at least one of $g(a^i), g(b^i)$ or $g(c^i)$ must be changed in order to make f a monotone conjunction. Thus, $\text{dist}_{\mathcal{D}_g}(g, \text{MCONJ}) \geq 1/3$ as promised.

4.1.3 The Strong Sampling Oracle

In the rest of the section, (f, \mathcal{D}) is drawn from either \mathcal{YES} or \mathcal{NO} . While each query to the sampling oracle returns a pair $(x, f(x))$, $f(x)$ is redundant given the definition of \mathcal{YES} and \mathcal{NO} : $f(x) = 0$ if $|\text{ZERO}(x)| = \ell$ and $f(x) = 1$ if $|\text{ZERO}(x)| = \ell/2$.

For clarity of the proof, we assume that T has access to a sampling oracle that sometimes returns extra information in addition to $x \sim \mathcal{D}$. Each time T queries, the oracle draws a string $x \sim \mathcal{D}$. Then

1. If $x = c^k$ for some $k \in [m]$, the oracle returns a pair (C_k, α_k) . (For the lower bound proof it is easier to work on sets instead of strings so we let the oracle return C_k instead of c^k . The extra information in the pair is the special variable index $\alpha_k \in C_k$.)
2. If $x = a^k$ or b^k for some $k \in [m]$ (the former happens only if $(g, \mathcal{D}_g) \sim \mathcal{NO}$), the oracle returns $(\text{ZERO}(x), \text{nil})$ (so no extra information for this case).

We will refer to this oracle as the *strong sampling oracle*. In the rest of the section we show that Lemma 4.1 holds even if T can make q queries to the strong sampling oracle and the black-box oracle each.

Let T be such an algorithm. Without loss of generality, we assume that T starts by making q queries to the strong sampling oracle. Let $Q = ((D_i, \gamma_i) : i \in [q])$ denote the sequence of q pairs that T receives in the sampling phase, where each pair $Q_i = (D_i, \gamma_i)$ has either $|D_i| = \ell/2$ and $\gamma_i = \text{nil}$ (meaning that D_i is A_k or B_k for some k) or $|D_i| = \ell$ and $\gamma_i \in D_i$ (meaning that D_i is C_k and γ_i is α_k for some $k \in [m]$). Let $\Gamma(Q)$ denote the set of integer γ_i 's in Q , i.e., α_k 's revealed in Q , $S(Q) \subset [n]$ denote $\cup_{i \in [q]} D_i$, and $I(Q) \subseteq [q]$ denote the set of $i \in [q]$ such that $|D_i| = \ell/2$.

4.2 Simulating T with No Access to the Black-Box Oracle

Our proof of Lemma 4.1 follows the high-level strategy of Glasner and Servedio [GS09]. We derive a new deterministic oracle algorithm T' from T that has *no access* to the black-box oracle. We then show that such an algorithm T' cannot distinguish the two distributions \mathcal{YES} and \mathcal{NO} (Lemma 4.2) but T' agrees with T most of the time (Lemma 4.3 and Lemma 4.9), from which Lemma 4.1 follows.

Now we define T' from T . In addition to a sequence Q of q samples, T' receives the set $R \subset [n]$ used in both procedures for \mathcal{YES} and \mathcal{NO} for free. Given R and Q , T' simulates T on Q as follows (note that T is not given R but receives only Q in the sampling phase): whenever T queries about $z \in \{0, 1\}^n$, T' does not query the black-box but passes the following bit $p(z, R, Q)$ back to T :

$$p(z, R, Q) = \begin{cases} 0 & \text{if } z_i = 0 \text{ for some } i \in [n] \setminus R \text{ or } i \in \Gamma(Q); \\ 1 & \text{otherwise.} \end{cases}$$

So T' receives R and makes q queries to the strong sampling oracle only.

The following lemma is the first step of our proof of Lemma 4.1.

Lemma 4.2. *Let T^* be any deterministic oracle algorithm that, on a pair (f, \mathcal{D}) drawn from \mathcal{YES} or \mathcal{NO} , receives R and a sequence Q of q samples but has no access to the black-box oracle. Then*

$$\left| \Pr_{(f, \mathcal{D}_f) \sim \mathcal{YES}} [T^* \text{ accepts}] - \Pr_{(g, \mathcal{D}_g) \sim \mathcal{NO}} [T^* \text{ accepts}] \right| = o(1).$$

Proof. We prove a stronger statement by giving the following extra information to T^* for free:

$$J = \left((C_i, \{A_i, B_i\}, \{\alpha_i, \beta_i\}) : i \in [m] \right).$$

Note that $\{A_i, B_i\}$ is given to T^* but they are not labelled. The same can be said about $\{\alpha_i, \beta_i\}$. Also R is revealed in J as $R = \cup_i C_i$. After J , T^* receives a sequence of q samples Q and now needs to either accept or reject with no other information about (f, \mathcal{D}) . We show that T^* cannot distinguish \mathcal{YES} and \mathcal{NO} .

By definition, the distribution of J when $(f, \mathcal{D}) \sim \mathcal{YES}$ is the same as that when $(f, \mathcal{D}) \sim \mathcal{NO}$, and we use \mathcal{J} to denote the distribution of J . Given a tuple J drawn from \mathcal{J} , we use \mathcal{Q}_J to denote the distribution of the sequence of q -samples Q conditioning on J when $(f, \mathcal{D}) \sim \mathcal{YES}$, and use \mathcal{Q}'_J to denote the distribution of Q conditioning on J when $(f, \mathcal{D}) \sim \mathcal{NO}$. We show that for any fixed J ,

$$\left| \Pr_{Q \sim \mathcal{Q}_J} [T^* \text{ accepts } (J, Q)] - \Pr_{Q \sim \mathcal{Q}'_J} [T^* \text{ accepts } (J, Q)] \right| = o(1). \quad (9)$$

The lemma then follows because procedures for \mathcal{YES} and \mathcal{NO} induce the same distribution \mathcal{J} of J .

For (9), it suffices to show that \mathcal{Q}_J and \mathcal{Q}'_J are close to each other. For this purpose, we say a sequence $Q = ((D_i, \gamma_i) : i \in [q])$ has *no collision* if no two sets D_i and D_j of Q come from $\{A_k, B_k, C_k\}$ with the same k . On the one hand, using the birthday paradox and our choices of q and m , $Q \sim \mathcal{Q}_J$ has a collision with probability $o(1)$. On the other hand, when Q has no collision, the probability of Q in \mathcal{Q}_J is exactly the same as that of Q in \mathcal{Q}'_J (which is a product of probabilities, one for each sample Q_i in Q : the probability of receiving each sample $Q_i = (D_i, \gamma_i)$ is $1/(6m)$ if $|D_i| = \ell$ and $1/(3m)$ if $|D_i| = \ell/2$). (9) follows, and this finishes the proof of the lemma. \square

4.3 Algorithms T' versus T When $(f, \mathcal{D}_f) \sim \mathcal{YES}$

Next, we show that T' agrees with T most of the time when (f, \mathcal{D}_f) is drawn from \mathcal{YES} , and when (f, \mathcal{D}_f) is drawn from \mathcal{NO} . We first deal with the easier case of \mathcal{YES} . We start with some notation.

Given a sequence of q -samples Q in the sampling phase, we use T_Q to denote the binary decision tree of T of depth q upon receiving Q . So each internal node of T_Q is labeled a query string $z \in \{0, 1\}^n$, and each leaf is labeled either accept or reject. Given Q , T walks down the tree by making queries about $f(z)$ to the black-box oracle. Given R and Q , T' walks down the same decision tree T_Q but does not make any query to the black-box oracle; instead it follows the bit $p(z, R, Q)$ for each query string z in T_Q .

We show that the probability of T' accepting a pair $(f, \mathcal{D}_f) \sim \mathcal{YES}$ is very close to that of T .

Lemma 4.3. *Let T be a deterministic oracle algorithm that makes q queries to the strong sampling oracle and the black-box oracle each, and let T' be the algorithm defined using T as in Section 4.2. Then*

$$\left| \Pr_{(f, \mathcal{D}_f) \sim \mathcal{YES}} [T \text{ accepts}] - \Pr_{(f, \mathcal{D}_f) \sim \mathcal{YES}} [T' \text{ accepts}] \right| \leq 0.1.$$

Proof. Given a sequence Q of q samples that T and T' receive in the sampling phase, we let \mathcal{YES}_Q denote

the distribution of (f, \mathcal{D}_f) drawn from \mathcal{YES} conditioning on Q . We claim that for any Q ,

$$\left| \Pr_{(f, \mathcal{D}_f) \sim \mathcal{YES}_Q} [T \text{ accepts}] - \Pr_{(f, \mathcal{D}_f) \sim \mathcal{YES}_Q} [T' \text{ accepts}] \right| \leq 0.1. \quad (10)$$

The lemma then follows directly. In the rest of the proof we consider a *fixed* sequence Q of samples.

We use $S = S(Q)$ to denote the union of sets in Q (so $|S| \leq q\ell = O(n/\log^3 n)$), and use $t = |\Gamma(Q)|$ to denote the number of α_i 's in Q . By the definition of \mathcal{YES} , every $\alpha_i \in S$ must appear in Q since \mathcal{D}_f has zero probability on strings a^i (so the only possibility of having an $\alpha_i \in S$ is because C_i is in Q , for which case α_i is also given in Q). Thus, there are exactly $m - t$ many α_i 's in $R \setminus S$ and we use Δ to denote the set of these α_i 's. Let \mathcal{R}_Q denote the distribution of the set R , conditioning on Q . Given an R from \mathcal{R}_Q , we abuse the notation and use $\mathcal{YES}_{Q,R}$ to denote the distribution of $(f, \mathcal{D}_f, \Delta)$, conditioning on Q and R .

We make a few simple but very useful observations. First the leaf of T_Q that T' reaches only depends on the set R it receives at the beginning; we use $w'(R)$ to denote the leaf that T' reaches. Second, conditioning on Q (and S), all indices $i \in [n] \setminus S$ are symmetric and are equally likely to be in R . Thus, in \mathcal{R}_Q , $R \setminus S$ is a subset of $[n] \setminus S$ of size $hr + 2m - |S|$ drawn uniformly at random. Finally, conditioning on Q and an R drawn from \mathcal{R}_Q , all indices $i \in R \setminus S$ are symmetric and equally likely to be in Δ (i.e., chosen as an α_i). In $\mathcal{YES}_{Q,R}$, Δ is a subset of $R \setminus S$ of size $m - k$ drawn uniformly at random.

Now we work on (10). Our plan is to show that, when $(f, \mathcal{D}_f) \sim \mathcal{YES}_Q$, most likely T and T' reach the same leaf of T_Q (and then either both accept or reject). We need a few definitions.

For each leaf w of T_Q , we define $H_w \subseteq [n] \setminus S$ to be the set of indices $i \in [n] \setminus S$ such that there exists a query string z on the path from the root to w but $z_i = 0$ and w lies in the 1-subtree of z . By the definition of H_w and the way T' walks down T_Q using R , a necessary condition for T' to reach w is that $H_w \subset R$. However, conditioning on Q , all indices $i \in [n] \setminus S$ are symmetric and equally likely to be in R drawn from \mathcal{R}_Q . So intuitively it is unlikely for T' to reach w if H_w is large.

Inspired by discussions above, we say a leaf w of T_Q is *bad* if $|H_w| \geq 0.02 \cdot n^{1/3}$; otherwise w is a *good* leaf (notice that whether w is good or bad only depends on Q (thus, S) and T_Q). We show that, when R is drawn from \mathcal{R}_Q , the probability of $w'(R)$ being bad is $o(1)$. To see this, for each bad leaf w of T_Q we have (letting $K = (n/2) + 2n^{2/3} - |S|$ be the size of $R \setminus S$ and plugging in $|S| \leq q\ell = O(n/\log^3 n)$)

$$\begin{aligned} \Pr_{R \sim \mathcal{R}_Q} [w'(R) = w] &\leq \Pr_{R \sim \mathcal{R}_Q} [H_w \subset R] = \frac{\binom{n-|S|-|H_w|}{K-|H_w|}}{\binom{n-|S|}{K}} \\ &= \frac{K - |H_w| + 1}{n - |S| - |H_w| + 1} \times \cdots \times \frac{K}{n - |S|} < 2^{-|H_w|} \leq 2^{-0.02 \cdot n^{1/3}}. \end{aligned}$$

By a union bound on the at most 2^q many bad leaves in T_Q and our choice of $q = O(n^{1/3}/\log^3 n)$ we have the probability of T' reaching a bad leaf is $o(1)$, when $R \sim \mathcal{R}_Q$. This allows us to focus on good leaves.

Let w be a good leaf in T_Q , and let R be a set from \mathcal{R}_Q such that $w'(R) = w$ (and thus, we must have $H_w \subset R \setminus S$). We bound probability of T not reaching w , when $(f, \mathcal{D}_f, \Delta) \sim \mathcal{YES}_{Q,R}$. We claim that this happens only when $\alpha_i \in H_w$ for some $i \in [m]$ (or equivalently, $H_w \cap \Delta$ is not empty).

We now prove this claim. Let z denote the first query string along the path from the root to w such that $f(z) \neq p(z, R, Q)$. By the definition of \mathcal{YES} and $p(z, R, Q)$, $p(z, R, Q) = 0$ implies $f(z) = 0$. As a result, we must have $f(z) = 0$ and $p(z, R, Q) = 1$. By $p(z, R, Q) = 1$, we have $\text{ZERO}(z) \subseteq R$ and $\text{ZERO}(z)$ has none of the α_i 's in $\Gamma(Q)$. By $f(z) = 0$, $\text{ZERO}(z)$ must contain an α_i outside of S , so this α_i is in $H_w \cap \Delta$. The latter is because $p(z, R, Q) = 1$ implies that z is one of the strings considered in the definition of H_w .

Using this claim, our earlier discussion on the distribution of Δ in $\mathcal{YES}_{Q,R}$ and $|H_w| < 0.02n^{1/3}$ as w

is a good leaf of T_Q , we have (letting $K = (n/2) + 2n^{2/3} - |S|$ be the size of $R \setminus S$)

$$\begin{aligned}
\Pr_{(f, \mathcal{D}_f, \Delta) \sim \mathcal{YES}_{Q,R}} [T \text{ does not reach } w] &\leq \Pr_{(f, \mathcal{D}_f, \Delta) \sim \mathcal{YES}_{Q,R}} [|H_w \cap \Delta| \neq \emptyset] \\
&= 1 - \frac{\binom{K-|H_w|}{m-t}}{\binom{K}{m-t}} \leq 1 - \left(1 - \frac{m}{K - |H_w| + 1}\right)^{|H_w|} \\
&\leq 1 - \left(1 - \frac{3m}{n}\right)^{|H_w|} \leq 1 - \left(1 - \frac{3}{n^{1/3}}\right)^{0.02n^{1/3}} \\
&\approx 1 - e^{-0.06} < 0.07.
\end{aligned}$$

Combining this and the fact that T' reaches a bad leaf with $o(1)$ probability, we have

$$\begin{aligned}
&\Pr_{(f, \mathcal{D}_f) \sim \mathcal{YES}_Q} [T \text{ and } T' \text{ reach different leaves of } T_Q] \\
&= \sum_w \sum_{R: w'(R)=w} \Pr_{(f, \mathcal{D}_f, \Delta) \sim \mathcal{YES}_{Q,R}} [T \text{ does not reach } w] \cdot \Pr_{\mathcal{R}_Q}[R] \\
&= o(1) + \sum_{\text{good } w} \sum_{R: w'(R)=w} \Pr_{(f, \mathcal{D}_f, \Delta) \sim \mathcal{YES}_{Q,R}} [T \text{ does not reach } w] \cdot \Pr_{\mathcal{R}_Q}[R] < 0.1.
\end{aligned}$$

This finishes the proof of (10) and the lemma. \square

4.4 Algorithms T' versus T When $(g, \mathcal{D}_g) \sim \mathcal{NO}$

We work on the more challenging case when $(g, \mathcal{D}_g) \sim \mathcal{NO}$. We start by introducing a condition on Q , and show that Q satisfies it with probability $1 - o(1)$.

Definition 4.4. Given a sequence $Q = ((D_i, \gamma_i) : i \in [q])$ of q samples from $(g, \mathcal{D}_g) \sim \mathcal{NO}$, we use H_i to denote the unique set C_k for some $k \in [m]$ that contains D_i . Then we say that Q is separated with respect to (g, \mathcal{D}_g) (since by Q itself one cannot tell if it satisfies the following condition) if for each $i \in [q]$ the number of $2 \log^2 n$ blocks of H_i that do not appear in any other H_j , $j \neq i$, is at least $(15/8) \log^2 n$.

Here is an observation that inspires (part of) the definition. Assume that algorithm T , given Q , suspects that D_i in Q is A_k for some k and wants to find α_k . However, indices that appear in D_i only, $D_i \setminus \cup_{j \neq i} D_j$, are symmetric and are equally likely to be α_k . Q being separated with respect to (g, \mathcal{D}_g) implies that there are many such indices in D . Of course the definition of Q being separated is stronger, and intuition behind it will become clear later in the proof of Lemma 4.9.

We show that when $(g, \mathcal{D}_g) \sim \mathcal{NO}$, Q is separated with respect to (g, \mathcal{D}_g) with probability $1 - o(1)$.

Lemma 4.5. When $(g, \mathcal{D}_g) \sim \mathcal{NO}$, a sequence Q of q samples from the sampling oracle is separated with respect to (g, \mathcal{D}_g) with probability $1 - o(1)$.

Proof. Recall that R' is the subset of R with α_i 's and β_i 's removed. Fix a $R' \subset [n]$ of size hr and a partition of R' into r pairwise disjoint blocks of size h each. We write J to denote the tuple consists of R' and blocks in R' , and \mathcal{NO}_J to denote the distribution of $(g, \mathcal{D}_g) \sim \mathcal{NO}$ conditioning on J . We also write C'_i to denote the set obtained from C_i after removing α_i and β_i . Given J , each C'_i is the union of $2 \log^2 n$ blocks drawn uniformly at random from the r blocks in R' .

Fix an J . Below we show that if each C'_i is the union of $2 \log^2 n$ random blocks and a sequence j_1, \dots, j_q is drawn from $[m]$ uniformly and independently, then with probability $1 - o(1)$ we have for each $i \in [q]$:

$$\text{the number of blocks of } C'_{j_i} \text{ that appear in } \cup_{k \neq i} C'_{j_k} \text{ is at most } \log^2 n / 16. \quad (11)$$

It follows that Q has the desired properties when $(g, \mathcal{D}_g) \sim \mathcal{NO}_J$ with probability $1 - o(1)$, and the lemma follows. For the rest of the proof we assume that J is fixed.

We now prove the claim. First of all by the birthday paradox and our choices of q and m , the probability of two indices j_1, \dots, j_q being the same is $o(1)$. Suppose that no two indices in j_1, \dots, j_q are the same. The distribution of $C'_{j_1}, \dots, C'_{j_q}$ is then the same as H_1, \dots, H_q , where each H_i is the union of $2 \log^2 n$ blocks in J drawn uniformly and independently at random. For the latter, we show that with probability $1 - o(1)$:

$$\text{for each } i \in [q], \text{ the number of blocks in } H_i \text{ that appear in } \cup_{k \neq i} H_k \text{ is at most } \log^2 n / 16. \quad (12)$$

This is not really surprising: on expectation, the number of blocks of H_i that also appear in $\cup_{k \neq i} H_k$ is

$$(q-1) \cdot \frac{2 \log^2 n \cdot 2 \log^2 n}{r} = o(1).$$

A formal proof that (12) happens with probability $1 - o(1)$ can be found in Appendix B. \square

We write E to denote the event that a sequence Q of q samples drawn from $(g, \mathcal{D}_g) \sim \mathcal{NO}$ is separated with respect to (g, \mathcal{D}_g) , and \mathcal{Q}_E to denote the probability distribution of Q conditioning on E . By definition not every Q is in the support of \mathcal{Q}_E ; we record the following property of Q in the support of \mathcal{Q}_E .

Property 4.6. *Given any $Q = ((D_i, \gamma_i) : i \in [q])$ in the support of \mathcal{Q}_E , each D_i has at most $\log n^2 / 8$ many blocks that appear in $\cup_{j \neq i} D_j$.*

Given a Q in the support of \mathcal{Q}_E , we write $\mathcal{R}_{Q,E}$ to denote the distribution of R , conditioning on Q and E . It is clear that $\mathcal{R}_{Q,E}$ is the same as \mathcal{R}_Q with E dropped since all indices in $[n] \setminus S(Q)$ remain symmetric and equally likely to be in R even given E .

Property 4.7. *For $R \sim \mathcal{R}_{Q,E}$, $R \setminus S(Q)$ is a set of size $hr + 2m - |S(Q)|$ drawn uniformly from $[n] \setminus S(Q)$.*

Given $Q = ((D_i, \gamma_i) : i \in [q])$, we use F_i to denote the other set of size $\ell/2$ paired with D_i , $i \in I(Q)$ (so F_i is A_k if D_i is B_k and vice versa). Given $Q = ((D_i, \gamma_i) : i \in [q])$ in the support of \mathcal{Q}_E and R in the support of $\mathcal{R}_{Q,E}$, we use $\mathcal{F}_{R,Q,E}^i$ to denote the distribution of F_i conditioning on R, Q and E . Then

Property 4.8. *Every F_i in the support of $\mathcal{F}_{R,Q,E}^i$ has at least $(7/8) \log^2 n$ blocks in $R \setminus S(Q)$. Moreover, they are drawn uniformly at random from blocks in $R \setminus S(Q)$. (More exactly, the number k of blocks of F_i in $R \setminus S(Q)$ is drawn from a certain distribution, where $k \geq (7/8) \log^2 n$ with probability 1, and then k blocks are drawn uniformly at random from blocks in $R \setminus S(Q)$.)*

We now show that T' agrees with T most of the time when $(g, \mathcal{D}_g) \sim \mathcal{NO}$:

Lemma 4.9. *Let T be a deterministic oracle algorithm that makes q queries to the strong sampling oracle and the black-box oracle each, and let T' be the algorithm defined using T as in Section 4.2. Then*

$$\left| \Pr_{(g, \mathcal{D}_g) \sim \mathcal{NO}} [T \text{ accepts}] - \Pr_{(g, \mathcal{D}_g) \sim \mathcal{NO}} [T' \text{ accepts}] \right| \leq 0.1.$$

Proof. Let Q be a sequence of q samples in the support of \mathcal{Q}_E . We prove that for any such Q :

$$\left| \Pr_{(g, \mathcal{D}_g) \sim \mathcal{N}\mathcal{O}_{Q,E}} [T \text{ accepts}] - \Pr_{(g, \mathcal{D}_g) \sim \mathcal{N}\mathcal{O}_{Q,E}} [T' \text{ accepts}] \right| \leq 0.09. \quad (13)$$

The lemma then follows from (13) and Lemma 4.5. Below we consider a *fixed* Q in the support of \mathcal{Q}_E .

For convenience, we let $S = S(Q)$, $\Gamma = \Gamma(Q)$ and $I = I(Q)$ since Q is fixed (so $|S| = O(n/\log^3 n)$). Given R in the support of $\mathcal{R}_{Q,E}$, we let $w'(R)$ denote the leaf of T_Q that T' reaches given R . We define H_w for each leaf w of T_Q and *good/bad* leaves of T_Q similarly as in the proof of Lemma 4.3. Using the same argument (as by Property 4.7, $R \setminus S$ is also drawn uniformly at random from $[n] \setminus S$) we have the probability of $w'(R)$ being bad is $o(1)$ when $R \sim \mathcal{R}_{Q,E}$. This again allows us to focus on good leaves in T_Q .

Now we fix a good leaf w of T_Q and a set R from $\mathcal{R}_{Q,E}$ with $w'(R) = w$. We use P_w to denote the path of query strings from the root to w . We drop R and Q in $p(z, R, Q)$ since they are fixed. In the rest of the proof we bound the probability of T not reaching w , when $(g, \mathcal{D}_g) \sim \mathcal{N}\mathcal{O}_{R,Q,E}$ (conditioning on R, Q, E).

We consider all the possibilities of T not reaching w . This happens because, for some z on the path P_w , $p(z) \neq f(z)$. By the definition of $\mathcal{N}\mathcal{O}$, at least one of the following four events holds. We bound the probability of each event by $o(1)$, when $(g, \mathcal{D}_g) \sim \mathcal{N}\mathcal{O}_{R,Q,E}$, and apply a union bound. For the four events below, Events E_0, E_1 and E_2 cover the case when $p(z) = 1$ but $f(z) = 0$. Event E_3 covers the case when $p(z) = 0$ but $f(z) = 1$ for some z in P_w . (Recall that $s = \log^2 n$.)

Event E_0 : There is a string z in P_w such that $p(z) = 1$ (so w is in the 1-subtree of z) but $z_{\alpha_k} = 0$ for some $\alpha_k \notin S$.

Event E_1 : There is a z in P_w such that $p(z) = 1$ but 1) $z_{\alpha_k} = 0$ for some $\alpha_k \in S$ and $\alpha_k \notin \Gamma$;
2) z is not k -special because there are more than $\log^2 n/4$ many blocks in A_k , each of which has at most s indices j with $z_j = 0$.

Event E_2 : There is a z in P_w such that $p(z) = 1$ but 1) $z_{\alpha_k} = 0$ for some $\alpha_k \in S$ and $\alpha_k \notin \Gamma$;
2) z is not k -special because there are more than $\log^2 n/4$ many blocks in B_k , each of which has (strictly) more than s indices j such that $z_j = 0$.

Event E_3 : There is a z in P_w such that $z_{\alpha_k} = 0$ for some $\alpha_k \in \Gamma$ but z is k -special, i.e., there are at least $3 \log^2 n/4$ blocks in A_k , each of which has (strictly) more than s indices j in it with $z_j = 0$; there are at least $3 \log^2 n/4$ blocks in B_k , each of which has at most s indices j in it with $z_j = 0$.

The probability of E_0 under $\mathcal{N}\mathcal{O}_{R,Q,E}$ is less than 0.07 by the same argument in the proof of Lemma 4.3.

Next we bound the probability of E_1 . Let $D'_i = D_i \setminus (\cup_{j \neq i} D_j)$ for each $i \in [q]$. Note that if there is an $\alpha_k \in S$ but $\alpha_k \notin \Gamma$, then $\alpha_k \in D'_i$ for some $i \in I$. Fixing a query string z in P_w and an $i \in I$, we bound the probability that E_1 happens at z and $\alpha_k \in D'_i$, and then apply a union bound on at most q^2 pairs of z and i .

Consider the scenario that D_i is indeed A_k for some k ; otherwise E_1 can never happen. When D_i is A_k , D'_i consists of $\{\alpha_k\}$ and $u \geq 7 \log^2 n/8$ blocks. (Note that u can be determined from the size of $|D'_i|$.) A key observation is that, conditioning on R, Q and E , all indices in D'_i are symmetric. So the choice of α_k as well as the partition of the rest of D'_i into u blocks are both done uniformly at random. Let $Z = \text{ZERO}(z) \cap D'_i$. By the observation above, part 1) of E_1 happens with probability $|Z|/|D'_i| = O(|Z|/\ell)$. So to make part 1) happen, one would like to set Z to be as large as possible. However, we claim that if $|Z| \geq 10 \log^4 n$, then with high probability, every block in D'_i has at least $2s$ indices in $\text{ZERO}(z)$, from which we know part 2) is violated because by E the number of blocks in $D_i \setminus D'_i$ is at most $\log^2 n/8$.

The claim above is not surprising, since each block by our discussion earlier is a subset of size h drawn from D'_i uniformly at random. So when $Z \geq 10 \log^4 n$, the expected number of indices of a block in Z is

$$|Z| \cdot \frac{h}{|D'_i|} \geq (10 \log^4 n) \cdot \frac{n^{2/3}}{2 \log^2 n} \cdot \frac{1}{n^{2/3} + 2} \geq 4 \log^2 n = 4s.$$

For a formal proof of the claim, we assume that blocks in D'_i are labelled: D'_i is partitioned into α_k and u blocks uniformly at random and then the blocks are labelled uniformly at random from 1 to u . Focusing on the block labelled j it is a set of size h drawn from D'_i uniformly at random and thus, can be also generated as a sequence of indices drawn from D'_i uniformly at random and independently until h distinct indices are sampled. However, even if we draw a sequence of h indices from D'_i uniformly at random and independently the probability of having at least $2s$ samples in Z is already $1 - n^{-\Omega(\log n)}$, e.g., by a folklore extension of Chernoff bound (see Lemma B.1). Thus, the probability of block j having at most $2s$ indices in $\text{ZERO}(z)$ is bounded by $n^{-\Omega(\log n)}$. By a union bound on all blocks in D'_i , we have that every block in D'_i has at least $2s$ indices in $\text{ZERO}(z)$ with probability $1 - n^{-\Omega(\log n)}$.

Combining the two cases when Z is small and large, we have that E_1 happens at a fixed z and D_i with probability $O(\log^4 n / \ell)$. Applying a union bound, E_1 happens with probability $O(q^2 \log^4 n / \ell) = o(1)$.

Next we consider E_2 . Let $Q = ((D_i, \gamma_i) : i \in [q])$, and F_i denote the set paired with D_i for each $i \in I$. A necessary condition for part 2) of E_2 to happen is that there exists an $i \in I$ such that more than $\log^2 n / 8$ blocks of F_i outside of S has more than s indices in H_w . To see this is the case consider a $z \in P_w$ and k such that E_2 happens at z and α_k . Then it must be the case that A_k is in Q and B_k is one of the F_i 's. By part 2) of E_2 , more than $\log^2 n / 4$ blocks of B_k has more than s indices in $\text{ZERO}(z)$. Given E , we know that at least $\log^2 n / 8$ many such blocks are outside of S , each of which has more than s indices in $\text{ZERO}(z)$. By $p(z) = 1$ z is one of the strings used to define H_w . Thus, all indices of $\text{ZERO}(z)$ outside of S belong to H_w .

We fix an $i \in I$ (and apply a union bound later). Also note that H_w is a fixed set in $R \setminus S$ of size at most $0.02n^{1/3}$ because w is a good leaf of T_Q . Consider any partition of $R \setminus S$ into blocks (and certain number of α_i 's and β_i 's). Then by the size of H_w , only $O(n^{1/3}/s)$ many of them can have an intersection of size more than s with H_w , and a block drawn uniformly at random from $R \setminus S$ is one such block with probability only $O(1/\log^4 n)$. By Property 4.8 $F_i \sim \mathcal{F}_{R,Q,E}^i$ draws at most $\log^2 n$ blocks uniformly at random from those in $R \setminus S$. The probability that more than $\log^2 n / 8$ of them have an intersection of size more than s with H_w can be bounded by $n^{-\Omega(\log^4 n)}$ (e.g., by following a similar argument used in Appendix B and considering a sequence of $2 \log^2 n$ blocks sampled uniformly and independently). By applying a union bound on all $i \in I$ we have that E_2 happens with probability $o(1)$ when $(g, \mathcal{D}_g) \sim \mathcal{N}_{R,Q,E}$.

For event E_3 we bound the probability that E_3 happens for some string z in P_w and some $\alpha_k \in \Gamma$, and then apply a union bound on at most q^2 many pairs of z in P_w and $\alpha_k \in \Gamma$. Consider an adversary that picks a string z and aims to make E_3 happen on z and α_k with probability as high as possible, given R, Q and E . Since $\alpha_k \in \Gamma$, C_k is a set in Q (paired with α_k as a sample Q_i). To ease the proof, we reveal β_k and all the blocks in C_k to the adversary for free and denote this information by J . Next, consider the distribution of A_k and B_k conditioning on J, R, Q and E . A key observation is that all blocks in J are equally likely to be in A_k and B_k : A_k is the union of α_k and $\log^2 n$ blocks drawn uniformly at random from J , and B_k is the union of β_k and the rest of blocks from J . This is because, given E and that C_k is in Q , neither A_k nor B_k is in Q . Thus, neither of J, R, Q, E reveals any information about how blocks in C_k are partitioned.

Let M denote the set of blocks in J that have more than s indices in $\text{ZERO}(z)$. For event E_3 to happen, A_k draws $\log^2 n$ blocks from J uniformly at random and have to hit $3 \log^2 n / 4$ blocks in M , while the rest can only have $\log^2 n / 4$ blocks in M , which is highly unlikely. For a formal proof, note that M must have at least $3 \log^2 n / 4$ blocks; otherwise the event never happens. Also, M certainly has at most

$2 \log^2 n$ blocks. We sample B_k using the following procedure: include in the first phase each block in B_k independently with probability $1/2$ and then either add or remove random blocks to make B_k with $\log^2 n$ blocks. By Chernoff bound, we have that with probability $1 - n^{-\Omega(\log n)}$ the first phase gets a B_k with at least $(11/32) \log^2 n$ blocks in M and at most $(33/32) \log^2 n$ blocks in total (since the expectation for number of blocks is between $3 \log^2 n/8$ and $\log^2 n$). When this happens, B_k sampled at the end must have at least $(5/16) \log^2 n > \log^2 n/4$ blocks in M .

Applying a union bound on all z in P_w and α_k in Γ , we have that E_3 happens with probability $o(1)$.

Combining these bounds on the probability of events E_i , $i \in \{0, 1, 2, 3\}$, we have the probability of T not reaching w when $(g, \mathcal{D}_g) \sim \mathcal{NO}_{R,Q,E}$ is less than 0.08. The lemma then follows. \square

4.5 Putting All Pieces Together

We now combine all the lemmas to prove Lemma 4.1.

Proof of Lemma 4.1. Let T be a deterministic oracle algorithm that makes at most q queries to each oracle, and T' be the algorithm that simulates T with no access to the black-box oracle. By Lemmas 4.2, 4.3, 4.9:

$$\left| \Pr_{(f, \mathcal{D}_f) \sim \mathcal{YES}} [T(f, \mathcal{D}_f) \text{ accepts}] - \Pr_{(g, \mathcal{D}_g) \sim \mathcal{NO}} [T(g, \mathcal{D}_g) \text{ accepts}] \right| \leq o(1) + 0.1 + 0.1 < 1/4.$$

This finishes the proof of Lemma 4.1 (and Theorem 1.2). \square

5 Extending the Upper Bound to General Conjunctions

In this section, we prove Theorem 1.3 using a simple reduction based on the following connection between MCONJ and CONJ. We need some notation. Given any $x \in \{0, 1\}^n$ and $C \subseteq [n]$, we use $x^{(C)}$ to denote the string obtained from x by flipping all coordinates in C . Given a probability distribution \mathcal{D} over $\{0, 1\}^n$, we use $\mathcal{D}^{(C)}$ to denote the distribution with $\mathcal{D}(x) = \mathcal{D}(x^{(C)})$ for all x .

Lemma 5.1. *Let \mathcal{D} be a probability distribution over $\{0, 1\}^n$, $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a Boolean function, and $x^* \in \{0, 1\}^n$ be a string such that $f(x^*) = 1$. Let $C = \text{ZERO}(x^*)$, and let $g : \{0, 1\}^n \rightarrow \{0, 1\}$ denote the Boolean function with $g(x) = f(x^{(C)})$ for all $x \in \{0, 1\}^n$. Then we have*

1. *If $f \in \text{CONJ}$, then $g \in \text{MCONJ}$.*
2. *If $\text{dist}_{\mathcal{D}}(f, \text{CONJ}) \geq \epsilon$, then $\text{dist}_{\mathcal{D}^{(C)}}(g, \text{MCONJ}) \geq \epsilon$.*

Proof. Assume that $f \in \text{CONJ}$. Then

$$f(x) = \left(\bigwedge_{i \in S} x_i \right) \wedge \left(\bigwedge_{i \in S'} \bar{x}_i \right),$$

where $S, S' \subseteq [n]$ are disjoint (since $f(x^*) = 1$). We also have that $C \cap S = \emptyset$ and $S' \subseteq C$. As a result,

$$g(x) = f(x^{(C)}) = \bigwedge_{i \in S \cup S'} x_i \in \text{MCONJ},$$

and the first part of the lemma follows.

We prove the contrapositive of the second part. Assume that $\text{dist}_{\mathcal{D}^{(C)}}(g, h) < \epsilon$, for some $h \in \text{MCONJ}$. Let h' denote the Boolean function with $h'(x) = h(x^{(C)})$. Then we have $h' \in \text{CONJ}$ and

$$\begin{aligned} \text{dist}_{\mathcal{D}}(f, \text{CONJ}) &\leq \text{dist}_{\mathcal{D}}(f, h') = \Pr_{x \in \mathcal{D}} [f(x) \neq h'(x)] \\ &= \Pr_{x \in \mathcal{D}} [g(x^{(C)}) \neq h(x^{(C)})] = \Pr_{x \in \mathcal{D}^{(C)}} [g(x) \neq h(x)] = \text{dist}_{\mathcal{D}^{(C)}}(g, h) < \epsilon. \end{aligned}$$

This finishes the proof of the second part of the lemma. \square

Now we prove Theorem 1.3.

Proof of Theorem 1.3. Given Lemma 5.1, a distribution-free testing algorithm for CONJ on (f, \mathcal{D}) starts by drawing $O(1/\epsilon)$ samples from \mathcal{D} to find a string x^* with $f(x^*) = 1$. If no such string is found, the algorithm accepts; otherwise the algorithm takes the first sample x^* with $f(x^*) = 1$ and runs our algorithm for MCONJ to test whether $g(x) = f(x^{(C)})$ is in MCONJ , where $C = \text{ZERO}(x^*)$, or g is ϵ -far from MCONJ with respect to $\mathcal{D}^{(C)}$, (Note that we can simulate queries on g using the black box for f query by query; we can simulate samples drawn from $\mathcal{D}^{(C)}$ using the sampling oracle for \mathcal{D} sample by sample.) and returns the same answer.

This algorithm is clearly one-sided given Lemma 5.1 and the fact that our algorithm for testing MCONJ is one-sided. When f is ϵ -far from CONJ , we have that $\mathcal{D}(f^{-1}(1)) \geq \epsilon$ because the all-0 function is in CONJ (when both x_i and \bar{x}_i appear in the conjunction for some $i \in [n]$). As a result, the algorithm finds an x^* with $f(x^*) = 1$ within the first $O(1/\epsilon)$ samples with high probability. It then follows from Lemma 5.1 that (f, \mathcal{D}) is rejected with high probability. \square

6 Extending the Lower Bound to General Conjunctions and Decision Lists

Let CONJ , DLIST and LTF denote the classes of all general conjunctions, decision lists, and linear threshold functions, respectively. Then we have $\text{MCONJ} \subset \text{CONJ} \subset \text{DLIST} \subset \text{LTF}$. In this section, we prove Theorem 1.4 for general conjunctions and decision lists. For this purpose we follow the same strategy used in [GS09] and prove the following property on the distributions \mathcal{NO} defined in Section 4.1:

Lemma 6.1. *With probability $1 - o(1)$, (f, \mathcal{D}_f) drawn from \mathcal{NO} satisfies $\text{dist}_{\mathcal{D}_f}(f, \text{DLIST}) \geq 1/12$.*

The same lower bound for CONJ and DLIST then follows directly from Lemma 4.1, given that $\text{MCONJ} \subset \text{CONJ} \subset \text{DLIST}$ and the fact that any pair (g, \mathcal{D}_g) drawn from \mathcal{YES} satisfies $g \in \text{MCONJ}$.

Proof of Lemma 6.1. Let (f, \mathcal{D}_f) be a pair drawn from \mathcal{NO} . Given any $i, j \in [m]$ such that $C_i \cap C_j = \emptyset$, we follow the same argument from Glasner and Servedio [GS09] to show that no decision list agrees with f on all of the following six strings $a^i, b^i, c^i, a^j, b^j, c^j$.

Assume for contradiction that a decision list h of length k :

$$(\ell_1, \beta_1), \dots, (\ell_k, \beta_k), \beta_{k+1},$$

agrees with f on all six strings. Let $\text{FIRST}(a)$ denote the index of the first literal ℓ_i in h that is satisfied by a string a , or $k+1$ if no literal is satisfied by a . Then we have

$$\min \{\text{FIRST}(a^i), \text{FIRST}(b^i)\} \leq \text{FIRST}(c^i) \quad \text{and} \quad \min \{\text{FIRST}(a^j), \text{FIRST}(b^j)\} \leq \text{FIRST}(c^j). \quad (14)$$

This is because by the definition of a^i, b^i and c^i , any literal satisfied by c^i is satisfied by either a^i or b^i . Next assume without loss of generality that

$$\text{FIRST}(a^i) = \min \{ \text{FIRST}(a^i), \text{FIRST}(b^i), \text{FIRST}(a^j), \text{FIRST}(b^j) \}. \quad (15)$$

By (14) we have that $\text{FIRST}(c^i) \geq \text{FIRST}(a^i)$. As $h(c^i) = f(c^i) = 0$ and $h(a^i) = f(a^i) = 1$, we have that $\text{FIRST}(c^i) \neq \text{FIRST}(a^i)$ and thus, $\text{FIRST}(c^i) > \text{FIRST}(a^i)$. This implies that the literal $\ell_{\text{FIRST}(a^i)}$ must be x_k for some $k \in B_i$. As $C_i \cap C_j = \emptyset$, we have $B_i \cap C_j = \emptyset$ and thus, $c_k^j = 1$. This implies that $\text{FIRST}(c^j) \leq \text{FIRST}(a^i)$, and $\text{FIRST}(c^j) < \text{FIRST}(a^i)$ because they cannot be the same given that $h(c^j) = f(c^j) = 0$ and $h(a^i) = f(a^i) = 1$. However, $\text{FIRST}(c^j) < \text{FIRST}(a^i)$ contradicts with (14) and (15).

As a result, when C_i and C_j are disjoint, one has to flip at least one bit of f at the six strings to make it consistent with a decision list. The lemma then follows from the fact that, with probability $1 - o(1)$, at least half of the pairs C_{2i-1} and C_{2i} , $i \in [m/2]$, are disjoint. \square

7 Extending the Lower Bound to Linear Threshold Functions

In this section we extend our lower bound to the distribution-free testing of linear threshold functions (LTF for short). We follow ideas from Glasner and Servedio [GS09] to construct a pair of probability distributions \mathcal{YES}^* and \mathcal{NO}^* with the following properties:

1. For each draw (f, \mathcal{D}_f) from \mathcal{YES}^* , f is a LTF;
2. For each draw (g, \mathcal{D}_g) from \mathcal{NO}^* , g is $(1/4)$ -far from LTFs with respect to \mathcal{D}_g .

Let $q = n^{1/3} / \log^3 n$. We follow arguments from the proof of Lemma 4.1 to prove the following lemma:

Lemma 7.1. *Let T be a deterministic algorithm that makes at most q queries to each oracle. Then*

$$\left| \Pr_{(f, \mathcal{D}_f) \sim \mathcal{YES}^*} [T(f, \mathcal{D}_f) \text{ accepts}] - \Pr_{(g, \mathcal{D}_g) \sim \mathcal{NO}^*} [T(g, \mathcal{D}_g) \text{ accepts}] \right| \leq \frac{1}{4}.$$

Our lower bound for LTFs then follows from Yao's minimax lemma. Below we define \mathcal{YES}^* and \mathcal{NO}^* in Sections 7.1 and 7.2, respectively, and prove Lemma 7.1 in Section 7.3.

7.1 The Distribution \mathcal{YES}^*

Recall the following parameters from the definition of \mathcal{YES} and \mathcal{NO} in Section 4.1:

$$\ell = n^{2/3} + 2, \quad m = n^{2/3}, \quad \text{and} \quad s = \log^2 n.$$

A draw (f, \mathcal{D}_f) from the distribution \mathcal{YES}^* is obtained using the following procedure:

1. Following the first five steps of the definition of \mathcal{YES} in Section 4.1.1 to obtain $R, C_i, A_i, B_i, \alpha_i, \beta_i$. For each $i \in [m]$, let a^i, b^i, c^i be the strings with $A_i = \text{ZERO}(a^i)$, $B_i = \text{ZERO}(b^i)$, $C_i = \text{ZERO}(c^i)$.
2. Define $u : \{0, 1\}^n \rightarrow \mathbb{Z}$ as following:

$$u(x) = 10n^2 \sum_{k \in [n] \setminus R} x_k + 5n \sum_{i \in [m]} x_{\alpha_i} - \sum_{k \in [n]} x_k.$$

Let $\theta = 10n^2(n/2 - 2m) + 5nm - (n - \ell/4)$.

- Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be the function with $f(x) = 1$ if $u(x) \geq \theta$, and $f(x) = 0$ otherwise. The distribution \mathcal{D}_f is defined as follows: we put $1/4$ weight on $\mathbf{1}^n$, and for each $i \in [m]$, we put $1/(2m)$ weight on b^i and $1/(4m)$ weight on c^i .

Clearly every pair (f, \mathcal{D}_f) drawn from \mathcal{YES}^* satisfies that f is an LTF. It is also easy to check that

$$f(a^i) = f(c^i) = f(\mathbf{1}^n) = 0 \quad \text{and} \quad f(b^i) = 1, \quad \text{for each } i \in [m].$$

7.2 The Distribution \mathcal{NO}^*

A draw (g, \mathcal{D}_g) from the distribution \mathcal{NO}^* is obtained in the following procedure:

- Following the definition of \mathcal{YES} in Section 4.1.1 to obtain $R, C_i, A_i, B_i, \alpha_i, \beta_i, c^i, a^i, b^i$.
- We follow the same definition of a string being *i-special* for some $i \in [m]$ as in Section 4.1.2. Let

$$J(x) = \{i \in [m] : x \text{ is } i\text{-special}\}, \quad \text{for each } x \in \{0, 1\}^n.$$

- Define $v : \{0, 1\}^n \rightarrow \mathbb{Z}$ as following:

$$v(x) = 10n^2 \sum_{k \in [n] \setminus R} x_k + 5n \left(|J(x)| + \sum_{i \in [m] \setminus J(x)} x_{\alpha_i} \right) - \sum_{k \in [n]} x_k.$$

Let θ be the same threshold: $\theta = 10n^2(n/2 - 2m) + 5nm - (n - \ell/4)$.

- Let $g : \{0, 1\}^n \rightarrow \{0, 1\}$ be the function with $g(x) = 1$ if $v(x) \geq \theta$, and $g(x) = 0$ otherwise. \mathcal{D}_g is defined as follows: we put $1/4$ weight on $\mathbf{1}^n$ and $1/(4m)$ weight on each of $a^i, b^i, c^i, i \in [m]$.

For each $i \in [m]$, we still have $g(c^i) = g(\mathbf{1}^n) = 0$, $g(b^i) = 1$ but $g(a^i)$ is flipped to 1 (since a^i is *i-special*). As $C_i = A_i \cup B_i$, we have that at least one of $g(a^i), g(b^i), g(c^i), g(\mathbf{1}^n)$ needs to be flipped to make g an LTF. It follows from the definition of \mathcal{D}_g that g is $(1/4)$ -far from LTFs with respect to \mathcal{D}_g .

7.3 Proof of Lemma 7.1

We follow arguments used in the proof of Lemma 4.1 to prove lemma 7.1.

Let T be any deterministic algorithm that makes q queries to each of the two oracles. We follow Section 4.1.3, and assume that T has access to the following *strong sampling oracle*:

- When the sampling oracle returns c^i for some $i \in [m]$, it returns the special index α_i as well;
- For convenience we also assume without loss of generality that the oracle always returns a sample drawn from the marginal distribution of \mathcal{D} within $\{a^i, b^i, c^i\}$ since samples of $\mathbf{1}^n$ are not useful in distinguishing \mathcal{YES}^* and \mathcal{NO}^* .

We show that Lemma 7.1 holds even if T receives q samples from the strong sampling oracle and makes q queries to the black-box oracle. We follow the same notation introduced in Section 4.1.3. Given a sequence $Q = ((D_i, \gamma_i) : i \in [q])$ of samples that T receives from the strong sampling oracle, let $\Gamma(Q)$ denote the set of integer γ_i 's in Q , let $S(Q) = \cup_{i \in [q]} D_i$, and let $I(Q)$ denote the set of $i \in [q]$ with $|D_i| = \ell/2$.

Next we follow Section 4.2 to derive from T a new deterministic oracle algorithm T' that has *no access* to the black-box oracle but receives R in addition to the sequence of samples Q at the beginning. We show

that T' cannot distinguish the two distributions \mathcal{YES}^* and \mathcal{NO}^* (Lemma 7.2), but T' agrees with T most of the time (Lemma 7.3 and Lemma 7.4), from which Lemma 7.1 follows.

The new algorithm T' works as follows:

Given R and Q , T' simulates T on Q as follows (note that T is not given R but receives only Q in the sampling phase): whenever T queries about $x \in \{0, 1\}^n$, T' does not query the oracle but computes

$$\phi(x) = 10n^2 \sum_{k \in [n] \setminus R} x_k + 5n (m - |I'(x)|) - \sum_{k \in [n]} x_k,$$

where $I'(x) = \text{ZERO}(x) \cap \Gamma(Q)$, i.e., the set of all α_i 's in $\Gamma(Q)$ revealed in the sampling phase such that $x_{\alpha_i} = 0$. T' then passes 1 back to T if $\phi(x) \geq \theta$ and 0 otherwise to continue the simulation of T . At the end of the simulation, T' returns the same answer as T .

Now we are ready to prove the three lemmas mentioned above.

The first lemma is to show that a deterministic oracle algorithm with no access to the black-box oracle cannot distinguish \mathcal{YES}^* and \mathcal{NO}^* distributions with high probability.

Lemma 7.2. *Let T^* be any deterministic oracle algorithm that, on a pair (f, \mathcal{D}) drawn from either \mathcal{YES}^* or \mathcal{NO}^* , receives R and a sequence Q of q samples but has no access to the black-box oracle. Then*

$$\left| \Pr_{(f, \mathcal{D}_f) \sim \mathcal{YES}} [T^* \text{ accepts}] - \Pr_{(g, \mathcal{D}_g) \sim \mathcal{NO}} [T^* \text{ accepts}] \right| = o(1).$$

Proof. The proof of the lemma is essentially the same as the proof of Lemma 4.2. The only difference here is that the distribution \mathcal{D} is also supported on $\mathbf{1}^n$. But because $\mathcal{D}_f(\mathbf{1}^n) = \mathcal{D}_g(\mathbf{1}^n) = 1/4$ in both \mathcal{YES}^* and \mathcal{NO}^* , the same proof works here. \square

Next we show that T' agrees with T most of the time when $(f, \mathcal{D}_f) \sim \mathcal{YES}^*$:

Lemma 7.3. *Let T be a deterministic oracle algorithm that makes q queries to the strong sampling oracle and the black-box oracle each, and let T' be the algorithm defined using T as above. Then*

$$\left| \Pr_{(f, \mathcal{D}_f) \sim \mathcal{YES}^*} [T \text{ accepts}] - \Pr_{(f, \mathcal{D}_f) \sim \mathcal{YES}^*} [T' \text{ accepts}] \right| \leq 0.1.$$

Proof. Fix a sequence Q of q samples. We prove the same statement conditioning on Q . Let \mathcal{R}_Q denote the distribution of the set R , conditioning on Q . We let T_Q denote the binary decision tree of T of depth q upon receiving Q , and let $w'(R)$ denote the leaf that T' reaches given R .

Following the same definition and argument used in the proof of Lemma 4.3 (as $\phi(x) < \theta$ if one of the variables outside of R is set to 0), it suffices to show for every R in the support of \mathcal{R}_Q such that $w = w'(R)$ is a *good* leaf (see the definition in the proof of Lemma 4.3), we have that T reaches w with high probability (conditioning on both Q and R). Note that $u(x)$ in the \mathcal{YES}^* distribution can also be written as:

$$u(x) = 10n^2 \sum_{k \in [n] \setminus R} x_k + 5n (m - |I(x)|) - \sum_{k \in [n]} x_k,$$

where $I(x)$ here is the set of all α_i 's, $i \in [m]$, such that $x_{\alpha_i} = 0$. Since $\phi(x) \geq \theta$, T does not reach w if and only if one of the strings x along the path from the root of T_Q to w satisfies

$$|I'(x)| < |I(x)| \quad \text{and} \quad \phi(x) \geq \theta > u(x).$$

Given that $\Gamma(Q)$ contains all α_i 's in $S(Q)$ (as a^i is not in the support of \mathcal{D}_f) it must be the case that $x_{\alpha_i} = 0$ for some $\alpha_i \notin S(Q)$ and thus, $\alpha_i \in H_w$ for some $i \in [m]$ (see the definition of H_w in the proof of Lemma 4.3). This is exactly the same event analyzed in the proof of Lemma 4.3, with its probability bounded from above by 0.1. This finishes the proof of the lemma. \square

Finally we show that T' agrees with T most of the time when $(g, \mathcal{D}_g) \sim \mathcal{NO}^*$:

Lemma 7.4. *Let T be a deterministic oracle algorithm that makes q queries to the strong sampling oracle and the black-box oracle each, and let T' be the algorithm defined using T as above. Then*

$$\left| \Pr_{(g, \mathcal{D}_g) \sim \mathcal{NO}^*} [T \text{ accepts}] - \Pr_{(g, \mathcal{D}_g) \sim \mathcal{NO}^*} [T' \text{ accepts}] \right| \leq 0.1.$$

Proof. Following Definition 4.4 and Lemma 4.5, the event E of Q being *separated* (with respect to (g, \mathcal{D}_g)) happens with probability $1 - o(1)$. Let \mathcal{Q}_E denote the probability distribution of Q conditioning on E . Fix a sequence Q in the support of \mathcal{Q}_E . Below we prove the statement of the lemma conditioning on both Q and E . Let $\mathcal{R}_{Q,E}$ denote the distribution of R conditioning on Q and E .

Similar to the proof of Lemma 4.9, it suffices to show that for every R in the support of $\mathcal{R}_{Q,E}$ such that $w = w'(R)$ is a good leaf, T reaches w with high probability, conditioning on R, Q and E .

Note that $v(x)$ from the \mathcal{NO}^* distribution can be also written as:

$$v(x) = 10n^2 \sum_{k \in [n] \setminus R} x_k + 5n(m - |I(x)|) - \sum_{k \in [n]} x_k,$$

where $I(x)$ is the set of all α_i 's, $i \in [m]$, such that $x_{\alpha_i} = 0$ and x is not i -special. Then T does not reach w only if for some x along the path from the root of T_Q to w , either $\phi(x) \geq \theta > v(x)$ or $v(x) \geq \theta > \phi(x)$.

When $\phi(x) \geq \theta > v(x)$, we have $|I(x)| > |I'(x)|$ and thus, one of the following two events must hold:

Event E_0^* : $\phi(x) \geq \theta$ (so w is in the 1-subtree of x) and $x_{\alpha_k} = 0$ for some $\alpha_k \notin S(Q)$;

Event $E_{1,2}^*$: $\phi(x) \geq \theta$, $x_{\alpha_k} = 0$ for some $\alpha_k \in S(Q)$ but $\alpha_k \notin \Gamma(Q)$, and x is not k -special.

For the case when $v(x) \geq \theta > \phi(x)$, we have $|I'(x)| > |I(x)|$ and thus, the following event must hold:

Event E_3^* : $x_{\alpha_k} = 0$ for some $\alpha_k \in \Gamma(x)$ and x is k -special.

Note that E_0^* is the same event as E_0 , $E_{1,2}^*$ is the same event as the union of E_1 and E_2 , and E_3^* is the same event as E_3 in the proof of Lemma 4.9. The lemma follows from bounds on their probabilities given in the proof of Lemma 4.9. \square

Lemma 7.1 then follows from Lemmas 7.2, 7.3, and 7.4.

References

- [AC06] N. Ailon and B. Chazelle, *Information theory in property testing and monotonicity testing in higher dimension*, Information and Computation **204** (2006), no. 11, 1704–1717.
- [AS05] N. Alon and A. Shapira, *Homomorphisms in graph property testing – a survey*, Electronic Colloquium on Computational Complexity (ECCC), Report **85** (2005), 281–313.
- [BFL91] L. Babai, L. Fortnow, and C. Lund, *Non-deterministic exponential time has two-prover interactive protocols*, Computational Complexity **1** (1991), no. 1, 3–40.

- [BLR93] M. Blum, M. Luby, and R. Rubinfeld, *Self-testing/correcting with applications to numerical problems*, Journal of Computer and System Sciences **47** (1993), no. 3, 549–595.
- [DR11] E. Dolev and D. Ron, *Distribution-free testing for monomials with a sublinear number of queries*, Theory of Computing **7** (2011), no. 1, 155–176.
- [Fis01] E. Fischer, *The art of uninformed decisions: A primer to property testing*, Science **75** (2001), 97–126.
- [GGR98] O. Goldreich, S. Goldwasser, and D. Ron, *Property testing and its connection to learning and approximation*, Journal of the ACM **45** (1998), no. 4, 653–750.
- [Gol98] O. Goldreich, *Combinatorial property testing (a survey)*, In: Randomization Methods in Algorithm Design, American Mathematical Society, 1998, pp. 45–60.
- [GS09] D. Glasner and R. Servedio, *Distribution-free testing lower bound for basic boolean functions*, Theory of Computing **5** (2009), no. 10, 191–216.
- [HK07] S. Halevy and E. Kushilevitz, *Distribution-free property-testing*, SIAM Journal on Computing **37** (2007), no. 4, 1107–1138.
- [HK08a] ———, *Distribution-free connectivity testing for sparse graphs*, Algorithmica **51** (2008), no. 1, 24–48.
- [HK08b] ———, *Testing monotonicity over graph products*, Random Structures & Algorithms **33** (2008), no. 1, 44–67.
- [RM99] R. Raz and P. McKenzie, *Separation of the monotone NC hierarchy*, Combinatorica **19** (1999), no. 3, 403–435.
- [Ron01] D. Ron, *Property testing (a tutorial)*, Handbook of Randomized Computing, Volume II (S. Rajasekaran, P.M. Pardalos, J.H. Reif, and J. Rolim, eds.), Springer, 2001.
- [RS96] R. Rubinfeld and M. Sudan, *Robust characterizations of polynomials with applications to program testing*, SIAM Journal on Computing **25** (1996), no. 2, 252–271.
- [Rub06] R. Rubinfeld, *Sublinear time algorithms*, available at <http://people.csail.mit.edu/ronitt/sublinear.html> (2006).
- [Val84] L.G. Valiant, *A theory of the learnable*, Communications of the ACM **27** (1984), no. 11, 1134–1142.

A Proof of Inequality (3)

We prove the last step of (3). Let $k = |B| = \Omega(rtW) \gg r$ since $W = \Omega(\epsilon)$. Let $\delta = 7/r$. Then

$$\begin{aligned} \frac{\binom{k}{r}}{\binom{k-\delta k}{r-1}} &= \frac{1}{r} \cdot \frac{(k - \delta k + 1)(k - \delta k + 2) \cdots k}{(k - \delta k - r + 2)(k - \delta k - r + 3) \cdots (k - r)} \\ &= \frac{k}{r} \cdot \frac{k - \delta k + 1}{k - \delta k - r + 2} \cdot \frac{k - \delta k + 2}{k - \delta k - r + 3} \cdots \frac{k - 1}{k - r} \\ &\leq \frac{k}{r} \cdot \left(\frac{k - \delta k + 1}{k - \delta k - r + 2} \right)^{\delta k - 1} \leq \frac{k}{r} \cdot \left(1 + \frac{2r}{k} \right)^{\delta k} = O\left(\frac{k}{r}\right). \end{aligned}$$

B Proof of (12)

We use the following folklore extension of the standard Chernoff bound:

Lemma B.1. *Let $p \in [0, 1]$ and X_1, \dots, X_n be a sequence of (not necessarily independent) $\{0, 1\}$ -valued random variables. Let $X = \sum_{i \in [n]} X_i$. If for any $i \in [n]$ and any $b_1, \dots, b_{i-1} \in \{0, 1\}$:*

$$\Pr[X_i = 1 \mid X_1 = b_1, \dots, X_{i-1} = b_{i-1}] \leq p,$$

then we have $\Pr[X \geq (1 + \delta) \cdot pn] \leq e^{-\delta^2 pn/3}$.

Now we prove (12). Fix an $i \in [q]$ and the $2 \log^2 n$ blocks in H_i . Then we sample all other $q - 1$ many H_j 's and bound the probability that (12) does not happen for i . We use the following procedure to sample H_j 's: for each $j \neq i$ sample a sequence of $4 \log^2 n$ blocks uniformly at random with replacement and set H_j to be the union of the first $2 \log^2 n$ distinct blocks sampled. This procedure, denoted by \mathcal{A} , fails if for some j , there are less than $2 \log^2 n$ distinct blocks from the $4 \log^2 n$ samples. When it succeeds, \mathcal{A} yields the desired uniform and independent distribution. We claim that \mathcal{A} succeeds with probability $1 - e^{-\Omega(r)}$.

To see this, for each j , its k th sample is the same as one of the previous $k - 1$ samples with probability at most $(k - 1)/r \leq 4 \log^2 n / r$, no matter what the outcomes of the first $k - 1$ samples are. By Lemma B.1, \mathcal{A} failed at H_j with probability $e^{-\Omega(r)}$ because this happens only if more than $2 \log^2 n$ samples have appeared before. By a union bound on j , \mathcal{A} succeeds with probability $1 - e^{-\Omega(r)}$.

Let U denote the union of all $(q - 1) \cdot (4 \log^2 n)$ blocks sampled by \mathcal{A} . Then

$$\begin{aligned} \Pr[(12) \text{ does not hold for } i] &\leq \Pr[U \text{ has } > \log^2 n / 16 \text{ blocks of } H_i \mid \mathcal{A} \text{ succeeds}] \\ &\leq \frac{\Pr[U \text{ has } > \log^2 n / 16 \text{ blocks of } H_i]}{\Pr[\mathcal{A} \text{ succeeds}]}. \end{aligned}$$

Using Chernoff bound, the probability of U having more than $\log^2 n / 16$ blocks of H_i is at most $n^{-\Omega(\log n)}$. (12) follows from $\Pr[\mathcal{A} \text{ succeeds}] \geq 1 - e^{-\Omega(r)}$ and a union bound on $i \in [q]$.